PKZIP for MVS
MVS/ESA, OS/390, & z/OS

User's Guide
PKMU-V5R5000

PKWARE, Inc.

PKWARE, Inc.

9009 Springboro Pike

Miamisburg, Ohio 45342

Sales: 937-847-2374

Support: 937-847-2687

Fax: 937-847-2375

Web Site: http://www.pkzip.com

Sales - E-Mail: pksales@pkware.com

Support - http://www.pkzip.com/support

5.5 Edition (2003)

***PKZIP for MVS*™, *PKZIP for OS/400*™, *PKZIP for VSE*™, *PKZIP for UNIX*™, and *PKZIP for Windows*™ are just a few of the many members in the PKZIP® family. PKWARE, Inc. would like to thank all the individuals and companies -- including our customers, resellers, distributors, and technology partners -- who have helped make PKZIP® the industry standard for Trusted ZIP solutions. PKZIP® enables our customers to efficiently and securely transmit and store information across systems of all sizes, ranging from desktops to mainframes.**

This edition applies to the following PKWARE of Ohio, Inc. licensed program:

***PKZIP for MVS*™ (Version 5, Release 5, 2003)**

PKZIP(R) is a registered trademark of PKWARE(R) Inc. Other product names mentioned in this manual may be a trademark or registered trademarks of their respective companies and are hereby acknowledged.

Any reference to licensed programs or other material, belonging to any company, is not intended to state or imply that such programs or material are available or may be used. The copyright in this work is owned by PKWARE of Ohio, Inc., and the document is issued in confidence for the purpose only for which it is supplied. It must not be reproduced in whole or in part or used for tendering purposes except under an agreement or with the consent in writing of PKWARE of Ohio, Inc., and then only on condition that this notice is included in any such reproduction. No information as to the contents or subject matter of this document or any part thereof either directly or indirectly arising there from shall be given or communicated in any manner whatsoever to a third party being an individual firm or company or any employee thereof without the prior consent in writing of PKWARE of Ohio, Inc.

Copyright. 2003 PKWARE of Ohio, Inc. All rights reserved.

Preface

The ***PKZIP®*** family of products consists of high performance data compression software. The archives resulting from compression by the PKZIP program can be transported or transmitted to other operating system platforms where they will undergo decompression by the PKUNZIP program or an acceptable substitute.

About this Manual

This manual provides the information needed to use ***PKZIP for MVS™*** in an operational environment. It is assumed that anyone using this manual has a good understanding of JCL and dataset processing. Note that the contents of this manual applies to the following operating systems:

- MVS/ESA 4.2 and above.
- OS/390 - all releases.
- z/OS - all releases.

- Chapter 1. An Introduction to PKZIP for MVS provides a general description of the PKZIP product, which is applicable to all supported platforms. This chapter also describes the MVS specific features of the PKZIP for MVS product and provides a simple description of how the PKZIP for MVS product is used to provide compression and decompression of datasets.
- Chapter 2. Provides more detailed examples of how specific file types should be processed by PKZIP for MVS. This chapter also details the new features and functions for PKZIP for MVS, Release 5.5.
- Chapter 3. Provides detailed information and guidance for customizing and fine tuning PKZIP specifically to your needs. In addition to the installation setup, this chapter explains licensing of PKZIP for MVS and provides you with the information to tailor your system configuration requirements.
- Chapter 4. Provides all of the general getting started information for invoking PKZIP and PKUNZIP. This chapter explains the details associated with compression, decompression, restrictions, migration, and an overview of PKZIP processing.
- Chapter 5. Provides a summary of the ZIP file processing procedures that include filtering, selection, requests, and the basic essentials for running PKZIP and PKUNZIP.
- Chapter 6. Explains ZIP files formats (text or binary) , files attributes, and file size considerations.
- Chapter 7. Provides information about the types of files that are supported by ***PKZIP for MVS™***, such as, sequential files, PDS, or PDSE members, and VSAM files.

- Chapter 8. Explains the three possible Archive states during processing. They are: old archive, temporary dataset, and new archive. Additionally, this chapter explains the functions of each of these formats.

- Chapter 9. Provides a comprehensive description of the commands and messages found in PKZIP for MVS.

- Chapter 10. Provides an overview of how to process GZIP files and archives, including information about GZIP restrictions and extensions.

- Chapter 11. Provides instructions on the use of other facilities provided with PKZIP for MVS, specifically the ISPF panel interface, to include setting options for configuration, defaults, and viewing archives.

- Chapter 12. Provides detailed information for invoking the ZIPAPI in assembler and COBOL.

- Glossary - Contains a list ***PKZIP for MVS™*** of terms.

- Index - Contains a detailed list and location of terms for this document.

Conventions Used in this Manual

Throughout this manual, the following conventions are used:

- The use of the `Courier` font indicates text that may be found in job control language (JCL), parameter controls, or printed output.

- The use of *italics* indicates a value that must be substituted by the user, for example, a dataset name. It may also be used to indicate the title of an associated manual or the title of a chapter within this manual.

- Bullets (•) indicate items (or instructions) in a list.

- The use of <angle brackets> in a command definition indicates a mandatory parameter.

- The use of [square brackets] in a command definition indicates an optional parameter.

- A vertical bar (|) in a command definition is used to separate mutually exclusive parameter options or modifiers.

Related Publications

Manuals relating to the ***PKZIP for MVS***™ product include:

- ***PKZIP for MVS***™ **User's Guide** - Provides detailed information on the ***PKZIP***® product set in MVS/ESA, OS/390, and z/OS operating environments. Also provided is a general introduction to data compression, PKZIP specific data compression, an overview on how to use PKZIP for MVS, PKZIP control cards, and parameters.

- ***PKZIP for MVS***™ ***& PKZIP for VSE***™ **Messages and Codes** - This provides information on the messages and codes that are displayed on the consoles, printed outputs, and associated terminals.

Other manuals within the ***PKZIP***® family of products include:

- ***PKZIP for OS/400***™ **User's Guide**

- ***PKZIP for VSE***™ **User's Guide**

- ***PKZIP for MVS***™ **Messages and Codes**

- ***PKZIP for Command Line - UNIX***™ **User's Guide**

- ***PKZIP for Command Line - Windows***™ **User's Guide**

Related IBM Publications

IBM Manuals relating to the ***PKZIP for MVS***™ product include:

- **System Codes** - Documents the completion codes issued by the operating system when it terminates a task or an address space. Describes the wait state codes placed in the program status word (PSW) when the system begins a wait state. Describes the causes of loops.

- **System Messages** - Documents the messages issued by the OS/390 operating system. The descriptions explain why the component issued the message, give the actions of the operating system, and suggest responses by the applications programmer, system programmer, and/or operator.

- **JES2 Messages** - Documents the messages issued by the JES2 subsystem. The descriptions explain why the component issued the message, give the actions of the operating system, and suggest responses by the applications programmer, system programmer, and/or operator.

- **JCL User's Guide** - Describes the job control tasks needed to enter jobs into the operating system, control the system's processing of jobs, and request the resources needed to run jobs. To perform

the tasks, programmers code job control statements. The user's guide assists in deciding how to perform job control tasks.

- **JCL Reference** - Describes the job control tasks needed to enter jobs into the operating system, control the system's processing of jobs, and request the resources needed to run jobs. To perform the tasks, programmers code job control statements. The reference guide; is designed to be used while coding the statements.

- **Access Methods Services** - Documents the functions that are available with Virtual Storage Access Method (VSAM) and describes the IDCAMS commands that can be issued to control VSAM datasets.

- **TSO/E Command Reference** - Documents the functions of the TRANSMIT and RECEIVE Command Facility used for the distribution and allocation of ***PKZIP for MVS™*** installation libraries.

Contents

PREFACE ..... **III**

About this Manual III
Conventions Used in this Manual IV
Related Publications V
Related IBM Publications V

CONTENTS ..... **VII**

CHAPTER 1. - AN INTRODUCTION TO PKZIP FOR MVS **1**

An Introduction to PKZIP 1
Data Compression 1
ZIP Archives 1
Cyclic Redundancy Check 2
Features Distinctive to PKZIP for MVS 2
Data Encryption 4
AES Key Sizes 5
Could "DES Cracker" like hardware break an AES key? 5
Monitoring Algorithm Security 5
What is the Life Expectancy of AES? 6
Cross Platform Compatibility 6

CHAPTER 2. - PKZIP FOR MVS™ 5.5 RELEASE INFORMATION **7**

Release Summary 7
New Features 7
New Commands 7
Command Changes 8
Message Changes 9
Enhancements for Secure Data 9
Restrictions for PKZIP for MVS Version 5.5 9
Environmental Execution Considerations 11
Region Size and Storage Usage 11
Reserved DDNAMEs 12
SYSPRINT 12
PKSPRINT 13
PKNODUMP 13
Use of System Utilities 13
SORT 13
Access Method Services 13
IEBGENER 13

CHAPTER 3. - INSTALLATION, LICENSING, AND CONFIGURATION **14**

Installation Overview 14
Type of Media Distribution for Installation 14

INSTALLATION FROM DOWNLOADED FILE OR CD 14
INSTALLING FROM 3480 OR 3490 TAPE 15
INITIALIZING THE LICENSE 16
Evaluation Period 16
Show System Information 16
Reporting the PKZIP for MVS 5.5 License 16
Applying a License Key or Authorization Code 17
PKZIP FOR MVS GRACE PERIOD 17
Running a Disaster Recovery Test 18
TAILORING SITE SPECIFIC CHANGES TO THE DEFAULTS MODULE 18
PROTECTING FILES WITH THE SAFETYEX MODULE 19
ACTIVATING THE ISPF INTERFACE 19
ISPF MAIN MENU 20

CHAPTER 4. - GETTING STARTED WITH PKZIP FOR MVS™ 21

INTRODUCTION TO PKZIP FOR MVS 21
INVOKING PKZIP OR PKUNZIP USING JCL 21
Notes for Invoking PKZIP or PKUNZIP Using JCL 22
Return Codes 22
COMPRESSING A DATASET 23
Notes for Dataset Compression 23
VIEWING THE CONTENTS OF AN ARCHIVE 24
Notes for Viewing the Contents of an Archive 24
–ACTION(VIEWDETAIL) 25
DECOMPRESSING A DATASET 26
Notes for Decompressing a Dataset 26
UPDATING OR REFRESHING A FILE 27
INVOKING PKZIP FOR MVS™ SERVICES 27
Invoking the PKZIP or PKUNZIP Programs From JCL (Batch or Started Task) 27
Invoking the PKZIP or PKUNZIP Programs as Called Programs Under TSO 27
Invoking ZIP or UNZIP TSO Command Line Interface 28
VALID ZIP ACTIONS 28
VALID ZIP OPTIONS 29
VALID UNZIP ACTIONS 29
Invoking the PKZIP for MVS™ ISPF Panel Interface 30
Invoking the PKZIP for MVS™ ZIP and UNZIP Services API 31
CONFIGURATION MANAGER 31
General Purpose 31
Making Changes to the Defaults 31
Assembling Your Changes 32
Inputs 32
User Input Sources (MVS) 32
Processing Order of Control Statements 32
CONFIGURATION MANAGER PROCESSING: MANAGING CONTROL STATEMENTS 33
Control Statement Definitions 33
TROUBLESHOOTING 33
PKZIP for MVS™ Messages 33

Debugging Controls 34
CHAPTER 5. FILE SELECTION AND NAME PROCESSING 35
ZIP PROCESSING FILE SELECTION 35
PRIMARY FILE SELECTION INPUTS 35
CATALOGED DATASET NAME FILTER REQUESTS 35
EXCLUSION FILTERS 35
INFILE DD REQUESTS 36
JES2 SYSIN INFILE SUPPORT 36
INPUT ZIP ARCHIVE FILES 36
FILE SELECTION PROCESSING NOTES 36
CATALOGED DATASET NAME AND INFILE REQUEST RESTRICTIONS 37
ZIP FILE NAMES 38
Summary of Commands Affecting ZIP Filename 38
ESSENTIALS FOR RUNNING PKZIP AND PKUNZIP 39
PKUNZIP 40
CHAPTER 6. - ZIP FILES 42
DATA FORMATS - TEXT OR BINARY 42
DATA FORMAT - TEXT RECORDS 43
DATA FORMAT - BINARY RECORDS 44
FILE ATTRIBUTES 44
LARGE FILE CONSIDERATIONS 44
DETERMINING FILE SIZE 45
CHAPTER 7. - FILE PROCESSING 46
FILE SUPPORT 46
Licensing 46
A. SEQUENTIAL FILES 47
Compressing Sequential Files 47
Extracting Records into a Sequential File 47
Managing a Sequential File ZIP Archive 48
Processing GDGs 48
File Concatenation for ZIP Processing 48
B. PDS AND PDSE MEMBERS 48
Selecting PDS Members for Compression 48
File Name or File Mask 49
DD Statements 49
Extracting Data into a PDS 49
Managing ZIP Archives as PDS Members 50
Load Libraries 50
Processing Individual Members 50
Load Module Control 50
Processing Entire Load Library 50
C. VSAM FILES 50
Compressing a VSAM File 51
VIEWDETAIL of a KSDS in an Archive 51

Extracting Data into a VSAM File 52
To Overwrite a Current VSAM File 52
To Restore a Compressed VSAM File 53
To Create a New VSAM File 53
Managing a VSAM ZIP Archive 53
To Update a VSAM ESDS ZIP Archive 54
To Process "Sparse" RRDS Files 54
Unsupported File Types 54
D. MAGNETIC TAPES AND CARTRIDGES 54
Copying a Tape-Based Archive to a Disk File 54
Compressing Data from Tape 55
Non-labeled Tapes (NL) 55
File Attributes 55
Extracting Data onto Tape 56
Managing a ZIP Archive on Tape 56
To Process Multiple-Volume Tape Archives 56
To Compress Data into a ZIP Archive on Tape 56
To View a Tape-Based Archive 57
To Extract Data From a Tape-Based Archive 57
To Update Files in a Tape-Based Archive 58

CHAPTER 8. - ZIP ARCHIVES 59

"OLD" ZIP ARCHIVE 59
"TEMPORARY" DATASET 60
"NEW" ZIP ARCHIVE 60

CHAPTER 9. - COMMANDS 61

COMMAND SYNTAX 61
FILE SELECTIONS VS. COMMANDS 62
&SYSUID 62
SUMMARY OF AVAILABLE COMMANDS 62
COMMAND DETAILS 74
Command Icon Legend 76
Notes for –ZIPPED_DSN 150
Defaults for –ZIPPED_DSN 151
NonVSAM files 151
VSAM Clusters for –ZIPPED_DSN 152

CHAPTER 10. - PROCESSING WITH GZIP 154

WHAT IS GZIP? 154
WHY USE GZIP? 154
PKZIP MVS™ IMPLEMENTATION NOTES FOR GZIP 155
GZIP Restrictions 155
GZIP Extensions 155
Processing GZIP Archives 156

CHAPTER 11. - USING THE ISPF INTERFACE 157

GETTING STARTED WITH THE ISPF INTERFACE 157
CONFIGURATION (OPTION 'C') 158
DEFAULTS (OPTIONS ZD AND UD) 158
Primary Commands 159
Changing Default Options 160
Including Changed Defaults 161
VIEW ARCHIVE (OPTION 'V') 161
Setting VIEW Options 161
Primary Commands 163
Line Commands 164
Display Fields 165
ZIP (OPTION 'Z') 166
SYSPRINT BROWSE (OPTION 'S') 167
MESSAGES (OPTION 'M') 168
LICENSE DISPLAY (OPTION 'L') 169
WHAT'S NEW (OPTION 'W') 169
CONTACT PKWARE (OPTION 'A') 170

CHAPTER 12. - USING THE APPLICATION PROGRAMMING INTERFACE (API) 171

INVOKING ZIPAPI IN ASSEMBLER 171
ZIPPARM Copy Member 172
INVOKING ZIPAPI IN COBOL 172
RETURN CODES 173

APPENDIX A - LICENSING REQUIREMENTS 174

LICENSED TYPES 174
PRODUCT FEATURES 175
LICENSING ENVIRONMENT 175
Evaluation Period 175
Current Use License 175
Reporting 176
Show System Information 177
Conditional Use 177

APPENDIX B - SAMPLE JOBSTREAMS 178

EXAMPLE 1: ZIP PDS TO AN ARCHIVE 178
EXAMPLE 2: ZIP PDS TO AN ARCHIVE 179
EXAMPLE 3: ZIP VSAM KSDS TO AN ARCHIVE 180
EXAMPLE 4: SUMMARY VIEW OF A DATASET 181
EXAMPLE 5: SUMMARY VIEW OF A DATASET 182
EXAMPLE 6: VIEW WITH DETAIL OF AN ARCHIVE 183
EXAMPLE 7: UNZIP AN ARCHIVE TO PDS 185
EXAMPLE 8: UNZIP AN ARCHIVE TO PDS 186
EXAMPLE 9: UNZIP AN ARCHIVE TO VSAM KSDS 187

APPENDIX C - INVOKING PKZIP/PKUNZIP FROM A PROGRAM 188

CALLZIPC SAMPLE ASSEMBLY SOURCE TO CALL PKZIP 190

CALLZIPA SAMPLE COBOL SOURCE TO CALL PKZIP 192
CALLZIPP SAMPLE PL/I SOURCE TO CALL PKZIP 193
CALLZIPR SAMPLE REXX SOURCE TO CALL PKZIP 194

APPENDIX D - 3480/3490 INSTALLATION JCL (COPYCART) 195

GLOSSARY 200

INDEX 212

Chapter 1. - An Introduction to PKZIP for MVS

An Introduction to PKZIP

PKZIP for MVS™ is a high performance data compression product, containing two main programs; PKZIP and PKUNZIP. The PKZIP program is used to compress or store files into a ZIP format archive, while the PKUNZIP program is used to extract data compressed by PKZIP. Processing control is available through the use of customized option modules, shared command lists, and individual job inputs. In addition to file selection, features such as compression levels and performance selections can be specified.

To guarantee data integrity, 32-bit Cyclic Redundancy Check (CRC) is a standard feature.

A ZIP archive is platform-independent; therefore, data compressed (zipped) on one platform, for example UNIX, can be decompressed (unzipped) on another platform, such as OS/390 or z/OS by using a compatible version of the PKUNZIP program.

Data Compression

Data compression reduces file size. A compressed data file uses less storage space and can be transferred faster. A data file to be compressed (a ZIP candidate) is compressed to a compact size (ZIPPED file). To use the file again, it must be uncompressed or extracted to its original size (UNZIPPED file).

A simple data compression method eliminates the repeating or redundant data, replacing it with representative information that will be used when restoring the data. An example of a simple data compression technique is the Run-Length Encoding method. This applies to redundant data where a repeating character, such as, the run is simply represented as a count or value, such as, the length. The compressed form is the repeated character with its count.

Example: B 2 2 2 2 E H H H H H H H H H

Compressed: B *4 2 E *9 H

However, to perform a thorough compression operation, more advanced algorithms and enhanced techniques are required. ***PKZIP for MVS™*** uses just such methods to achieve maximum results.

ZIP Archives

PKZIP for MVS™ stores compressed data files into ZIP archives. There is no limit to the number of archives you may create.

- A ZIP archive refers to any valid ZIP-format file created by a ***PKZIP®*** 2.x-compatible product.
- Each archive can store up to 65,535 files.
- Files up to 4 gigabytes in size can be compressed into a standard PKZIP Archive. (See Chapter 10. - Processing with GZIP. For information about compressing files larger than 4 gigabytes).
- Each archive may contain up to 4 gigabytes of compressed data.
- For each file in the archive, the following information is stored with the compressed data:

- Filename.
- File directory date and time.
- File's initial CRC value. See Cyclic Redundancy Check.
- Method of compression used.
- ***PKZIP for MVS™*** Version required for file extraction.
- File size, uncompressed.
- File size, compressed.

- Some files may contain the following additional information:
  - The version of ***PKZIP for MVS™*** that created the file.
  - File attributes.
  - Any comment about the file.
  - A comment about the archive.
  - Platform specific attributes (see Cross Platform Compatibility).

Cyclic Redundancy Check

Cyclic Redundancy Check (CRC) is a method used to check the integrity of the data file when it is restored from the ZIP archive.

While a file is compressed, a ***PKZIP for MVS™*** algorithm computes a 32-bit hexadecimal value for its data. That CRC value is stored with the file in the ZIP archive. The data in the file is extracted, ***PKZIP for MVS™*** processes it again by the same algorithm to produce a second CRC value. Note that if the data is the same as its previous state, the same CRC value will be produced. The two CRC values are compared, and if the extracted value does not match the initial stored value, the integrity of the file is in question and ***PKZIP for MVS™*** reports the results. It is possible that the data was corrupted within the ZIP Archive during file transfer operations.

Features Distinctive to PKZIP for MVS

Features distinctive to the z/OS, OS/390, and MVS/ESA operating environments include the following:

Process execution from ISPF Panels, as a TSO command processor, from a user's program (for example, ASSEMBLER, COBOL) or batch JCL.

A robust ISPF panel interface provides ZIP Archive directory displays in a table format, and allows for individual ZIPPED File selection: Browse, View, Extract, and Delete.

Compressing and extracting of datasets of the following types on DASD:

Sequential files.

PDS and PDSE members.

VSAM files (KSDS, ESDS, RRDS).

JES2 subsystem input files (for example, `//ddname DD *`).

Command Extensions allowing greater flexibility in file selection.

Unique filename translation to/from system/390 DSNAME conventions and the UNIX-style names typically found in zip archives.

Compressing and extracting of datasets of the following types on Tape or Cartridge:

Sequential files.

Compressing and extracting of files to OS/390 AND MVS/ESA Load Libraries.

Compressing and extracting of files to Generation Data Groups (GDGs).

GDG files can be used as a ZIP Archive.

Retention of dataset allocation information, such as dataset organization, device type, and DCB/Cluster attributes. Preservation of this information will allow for duplication of the file with the same characteristics during the UNZIP process. Support of ZIP Archives within the following dataset organizations:

Sequential files (DASD, Tape, or Cartridge).

PDS and PDSE members.

VSAM ESDS.

PKZIP for MVS™ selects datasets for processing based upon user-specified control statements, DD JCL specifications, or user-defined filtering lists.

PKZIP for MVS™ executes on MVS/ESA and higher.

PKZIP for MVS™ runs in AMODE 31 and primarily uses storage above the 16-Mb line. However, certain operating system control blocks and system services require virtual storage below the 16-Mb line. The amount of virtual storage available within each of these areas of an address space will limit the use of some performance options (for example, multi-tasking and temporary files in storage) and capabilities.

Offers customizable defaults during installation. Multiple defaults modules may be created for use in a variety of application needs.

Allows for the use of pre-defined command files saved in a place selected by the user or system's administrator. These can be referenced by multiple jobs or users, thus eliminating the need for individual JCL command streams, or used in combination with individual job inputs to provide a consistent set of processing controls.

Various features of ***PKZIP for MVS™*** are individually licensed (see Appendix A - Licensing Requirements).

Data Encryption

PKZIP for MVS™ Version 5.5 can encrypt data for security control and provide a password lockout for extracting data. Varying security levels are available with 96, 128, 192, and 256 bit encryption, with two encryption algorithms. As of July 1, 2001 the US Government mandated through the Gramm-Leach-Bliley Act that customer information must be encrypted to keep it confidential. (See Section on Advanced Encryption Standard (AES) for more information to meet this government requirement for advanced encryption of sensitive data). ***PKZIP for MVS™*** Version 5.5 implements the Advanced Encryption Standard.

Decryption requires a key. ***PKZIP for MVS™*** Version 5.5 uses a multi-layer key generation process (based on a user-specified password of up to 200 characters) that creates a unique internal key for each file being processed. In addition, the same password will result in a different system generated key for each file.

PKZIP for MVS™ Version 5.5 implements the use of Cipher Block Chaining (CBC) to further enhance industry standard encryption algorithms. This feature ensures that each block of data is uniquely modified, further protecting the data from fraudulent access.

The following matrix illustrates cross-product compatibility of encrypted data and general encryption options available on the operating systems supported by PKZIP.

More specifically, 96 bit password encryption is compatible and can be ported to and from any of the platforms listed below. Advanced password encryption is also available in a cross platform environment that supports the AES (AES is the Government requirement for password and data encryption as of May 26, 2002) requirement of 128, 192, and 256 bit password encryption.

SECURITY TYPES	Generic ZIP Utilities	PKZIP for Windows	PKZIP for OS/400	PKZIP for OS/390	PKZIP for VSE	PKZIP for MVS (See Note 1below)	PKZIP for UNIX (See Note 2 below)
96 bit Encryption	**X**	**X**	**X**	**X**	**X**	**X**	**X**
This is advanced encryption; that is compatible with Windows.							
PKWARE Certificate Based Encryption		**X**					**X**
This is advanced encryption; that is compatible with multiple platforms.							
128, 192, & 256 bit AES Password Encryption		**X**	**X**	**X**		**X**	**X**

Note 1: *PKZIP for MVS*™ Version 5.5 supports the following operating systems: MVS/ESA 4.2 and above, OS/390, and z/OS.

Note 2: PKZIP supports advanced password encryption for the following types of UNIX: Sun Solaris 2.6 and above, HP-UX 10.20 and above, IBM AIX 4.3 and above, Intel Linux based on 2.4 kernel.

***PKZIP for MVS*™** uses AES which is the official US Government standard for encryption. The AES algorithm was approved as the Federal Information Processing Standard by the Commerce Department on May 26th, 2002.

The Rijndael (the name combination of the two researchers who developed Rijndael, Dr. Joan Daeman and Dr. Vincent Rijmen) algorithm uses a combination of advanced security, performance, efficiency, ease of implementation, and flexibility to make it the appropriate standard of advanced encryption for the AES.

Rijndael performs consistently in both hardware and software and in cross platform environments regardless of its use in feedback or non-feedback modes. Rijndael's key setup time is very good, and its key agility is excellent. Memory requirements are very low, making it the first choice for restricted-space environments, in which it also demonstrates high performance. Power and timing attacks are easily defended against due to Rijndael's operations.

Note that the AES was intentionally developed to replace DES.

AES Key Sizes

Currently, AES has three key sizes. They are: 128, 192, and 256 bits. Key sizes are depicted in the following decimal terms:

3.4×10^{38} possible 128-bit keys;

6.2×10^{57} possible 192-bit keys; and

1.1×10^{77} possible 256-bit keys.

In comparison, DES keys are only 56 bits, which means there are approximately 7.2×10^{16} possible DES keys. Therefore, there they are on the order of 10^{21} times more AES 128-bit keys than DES 56-bit keys.

Could "DES Cracker" like hardware break an AES key?

Specialized "DES Cracker" machines were built in the late 1990's that could recover a DES key after only a few hours. By trying possible key values, the hardware could determine which key was used to encrypt a message.

To illustrate the higher level of security that AES provides versus DES, if a machine that could recover a DES key in a second, such as, try 2^{55} keys per second, then it would take that machine approximately 149 thousand-billion (149 trillion) years to crack a 128-bit AES key. A further perspective is that the universe is believed to be less than 20 billion years old, thus making the case that cracking a 128-bit AES key is nearly impossible with today's technology.

Monitoring Algorithm Security

The National Institute of Standards and Technology (NIST) continues to follow developments in the cryptanalysis of Rijndael. The AES is formally reevaluated every five years. Plans for maintenance activities for the standard will be developed in the future, with full consideration of all circumstances. When

an issue arises that requires immediate attention, NIST will act expeditiously and consider all available alternatives.

What is the Life Expectancy of AES?

No one can be certain of how long the AES will remain secure. However, NIST's DES was the U.S. Government standard for almost twenty years before it was "cracked" by a massive parallel network computer attacks and special-purpose "DES-cracking" hardware. The AES supports significantly larger key sizes than that of DES. Barring any attacks against AES that are faster than key exhaustion, and the advent of future advances in technology, AES could remain secure well beyond twenty years.

Cross Platform Compatibility

Cross platform compatibility provides ***PKZIP for MVS™*** its ability to allow data to move between different computer operating environments. PKZIP was intentionally designed for cross platform use. Regardless of platform, ***PKZIP for MVS™*** archives are compatible with ***PKZIP for OS/400™***, ***PKZIP for VSE™***, ***PKZIP for UNIX™***, ***PKZIP for LINUX™***, ***PKZIP for DOS™***, and ***PKZIP for Windows™*** to name a few. Because ***PKZIP for MVS™*** automatically converts the data between EBCDIC and ASCII, files prepared on the host are readable on any PC or UNIX system. The internal format of a ZIP archive is identical regardless of which platform compressed the files that the archive contains. If you want to transfer data across platforms using any other version of PKZIP or other ZIP utility, you should always run a test to verify the cross platform compatibility.

PKZIP for MVS™ uses the same ZIP file archive format used by other ***PKZIP®*** compatible products, independent of the platform on which it is running. ***PKZIP for MVS™*** archives are not platform dependent allowing greater flexibility in file usage. Data can be zipped on one platform, for example UNIX, and unzipped onto another platform, such as OS/400. To do this, ***PKZIP for MVS™*** converts the data structure into the ***PKZIP®*** format and saves the appropriate file information in the ZIP Archival directory entries.

If you want to transfer data across platforms using any other ”Zip compatible” product, the user should check with the supplier first to confirm that the versions of PKZIP that are compatible.

Chapter 2. - PKZIP for MVS™ 5.5 Release Information

Release Summary

New Features

New features for ***PKZIP for MVS™*** Release 5.5:

- Advanced Encryption
- Improved Compression
- Ehanced File Filtering Capabilities
- PASSWORD echo masking
- Add –ACTION(COPY)
- Add –CHECK_SYSIN_MEMBER command
- Add –ENCRYPTION_METHOD command
- Add –EXCLUDE command
- Add –KEY_PROTECT_LEVEL command
- Add –PKSUPPRC command
- Add –PRESERVE_CMD_SPACE command
- Rebuilt Messages Manual
- DOC Memory Usage Info
- DOC Abend S213-30 (IEC143I) when competing with UNZIP to PDS
- PANVALET Subsystem Support for command input

Note: Some commands were initially introduced during the interim release with higher levels of 5.0 and in the 5.5 Release.

New Commands

The Default values for the following commands are new. Be sure to review these commands to familiarize yourself with the new functionality these commands provide for ***PKZIP for MVS™*** Version 5.5.

Command	Description	Values
–EXCLUDE	Enhanced file filtering capabilities.	User-supplied
–CHECK_SYSIN_MEMBER	Verifies a command input stored in a PDS or PDSE member.	Y|N
–KEY_PROTECT_LEVEL	Specifies a relative intensity of encryption key protection.	1 / 2
–PRESERVE_CMD_SPACE	Preserves or removes blanks proceeded by a "|".	Y|N

Command	Description	Values
–PKSUPPRC	Allows the return code to be suppressed on certain conditions.	ZPAM092E - Nothing to do. ZPAM093W - No Files match: Initializing/Copying Archive. ZPEX013 - Truncation.
–SUPPRESS_DYNALLOC_MSGS	Specifies that the dynamic allocation messages in job log be suppressed.	–NODYNMSGS
–DATA_TYPE(DETECTX)	Provides automatic detection and translation of ASCII text during UNZIP processing (similar to DETECT for ZIP processing).	Default remains as "DETECT".

Command Changes

The default values for the following commands have been changed. When assembling an existing installation defaults module (ACZDFLT), these values should be reviewed for applicability and adjusted, as required. **Upgrade note:** Installations suppressing the //SYSIN PDS member verification for performance reasons with PROC_OPT1=N (available with 5.0.10 maintenance) in ACZDFLT should change to –CHECK_SYSIN_MEMBER=N in the assembly of ACZDFLT. PROC_OPT1 will no longer be used for this purpose in Release 5.5 and above.

Command	Old Values	New Values
–ARCHIVE_DIR_BLOCKS	10	56
–ARCHIVE_SPACE_PRIMARY	100	10
–ARCHIVE_SPACE_SECONDARY	100	10
–ARCHIVE_SPACE_TYPE	TRK	CYL
–ARCHIVE_UNIT	SYSALLDA	SYSDA
–COMPRESSION_LEVEL	NORMAL	SUPERFAST
–MULTI_THREAD_LIMIT	1	3
–OUTFILE_SPACE_TYPE	TRK	CYL
–OUTFILE_SPACE_PRIMARY	100	10
–OUTFILE_SPACE_SECONDARY	100	10
–OUTFILE_UNIT	SYSALLDA	SYSDA
–PASSWORD		Increased Maximum length to 200 characters.
–PARMLIB_DSNAME_ZIP		NULLFILE
–PARMLIB_DSNAME_UNZIP		NULLFILE
–PROCESS_ALIAS	N	Y
–SAVE_FILE_ATTRIBUTES	BOTH	CENTRAL
–TEMP_UNIT	NULL	SYSDA
–VSAM_SPACE_PRIMARY	100	10

Command	Old Values	New Values
–VSAM_SPACE_SECONDARY	100	10
–VSAM_SPACE_TYPE	TRK	CYL

Message Changes

The table below identifies new and changed messages for ***PKZIP for MVS™*** Version 5.5. Be sure to review this table before using ***PKZIP for MVS™*** Version 5.5.

Message ID number	New	Changed
ZPAM010I		X
ZPAM082W		X
ZPAM255I		X
ZPAM291I		X
ZPAM292I		X
ZPEN001I	X	
ZPEN002W	X	
ZPEN003W	X	
ZPEN004E	X	
ZPCM019E	X	
ZPCM203E	X	
ZPEX082E	X	
ZPEX083I	X	

Enhancements for Secure Data

- New Encryption Method command.
- The password will no longer be echoed in the SYSPRINT stream. The value '-PASSWORD(**********)" will be displayed instead.
- When entering passwords on the ISPF panels, the input field has been changed to non-display. A password verification field has been added on the password prompting screens to assist you in verifying that the correct password has been entered. The password may be displayed by selecting a panel option.

Restrictions for PKZIP for MVS Version 5.5

The following restrictions apply to PKZIP for MVS Version 5.5:

In environments that do not use the Integrated Catalog Facility (ICF), PKZIP cannot function fully. It is unable to rename the temporary dataset it creates as a ZIP archive to the name specified by you. The integrity of the ZIP archive is not impaired in any way and archived files can be extracted successfully. However, the temporary dataset name of the ZIP archive should be changed to the name required by you after PKZIP has completed.

When two (or more) files from a ZIP archive are extracted with the same MVS dataset name, the last file will overwrite any previous file(s).

PKZIP for MVS will not process datasets that are spread over more than 31 volumes.

Extracting to a GDG dataset via –OUTFILE_DD will result in the use of the user-specified DCB values. The user must ensure that these values are appropriate to the record lengths being written.

The number of files or PDS members that can process in one operation may be restricted by the number of concurrent DD's that can be used in the address space, such as, the size of the TIOT. For further information on this limit, see the documentation for DD statements in the IBM *JCL User's Guide*.

Some IDCAMS DEFINE Cluster options can be specified at the Cluster and Data (and Index if appropriate) levels. However, a few of these options, when specified using ARCH* or OUT* commands during PKZIP or PKUNZIP operations, will set only the Data (and Index) components. This is because some PKZIP ARCH* and PKUNZIP OUT* commands which apply to Cluster, Data, and Index components, currently set both the data and index attributes, and ignore the Cluster level component. These may in future, set the Cluster level option only. Commands that may change in this way are shown in the following table. For these commands, it is recommended that the ARCHDATA* and ARCHINDX, or OUTDATA* and OUTINDX* options be used.

PKUNZIP Command	Comments
–ARCHEEXT	Is effectively the same as setting both –ARCHDATAEEXT and –ARCHINDXEEXT.
–ARCHOWNER	Is effectively the same as setting both –ARCHDATAOWNER and –ARCHINDXOWNER.
–OUTEEXT	Is effectively the same as setting both –OUTDATAEEXT and –OUTINDXEEXT.
–OUTOWNER	Is effectively the same as setting both –OUTDATAOWNER and –OUTINDXOWNER.

When processing Tape datasets, without a Tape license, PKZIP and PKUNZIP may request that a Tape be mounted, prior to checking that the product is not actually licensed to process the Tape. In this circumstance, the Tape mount must be satisfied before PKZIP for MVS processing will proceed, even when this processing will just inform you that it is not possible to process the Tape.

PDS members containing positioning information (for example Load members with Overlay sections) are not supported. In certain circumstances these might be processed with unpredictable results.

PDSE program objects are not currently supported in native format. IEBCOPY should first be used to offload the PDSE Library to a sequential file and the resulting sequential file can be Archived. Subsequently, after extracting the unloaded version of the PDSE, it can be reloaded with IEBCOPY.

GZIP (GNU zip) file processing has a number of restrictions as documented in Chapter 10.

Dataset Alias entries can be used to select datasets, however, the true name will be used to process filename associations in the archive. The Dataset Alias name is not retained.

Values for dynamic allocation requests by ***PKZIP for MVS™*** may be added, altered, or removed by installation-dependent storage management services, for example, DF/SMS. Allocation results may be different from those specified by ***PKZIP for MVS™*** commands or default values.

PKZIP for MVS™ makes use of Access Method Services User I/O Routines for SYSIN and SYSPRINT file requests. OEM products and/or Installation-written routines that modify standard IBM processing for these exits should not be active during PKZIP processing.

The OS/390 - dependent data types, such as binary load modules, may not be usable on other platforms. That is, ***PKZIP for MVS™*** does not convert executable programs from one system platform to another.

Although it is possible for Archives to be appended to other Archives in a dataset during a ZIP process, for example, DISP(=MOD,CATLG, in MVS, or using the UNIX append">>" operator for files), this is not recommended. The result is that 'dead' Archives will be carried along in the file, and various ZIP products will read the file differently, some look for the ZIP Archive directory structure from the beginning, others from the end of the file.

PKZIP for MVS™ attempts to read the first archive found from the beginning of the file, for performance reasons and to perfom an Archive integrity check. If an inconsistency in the initial header structures exist, a secondary search from the back of the archive will be attempted. ***PKZIP for MVS™*** will accept up to 64k of non-Archive data at the end of the Archive file when searching for the end of the directory (from the back). This limit does not apply when the local directory structure is intact.

For more information regarding Data Formats, see Chapter 6. - Text or Binary for a discussion regarding special considerations when transferring files between different platform types.

PKZIP for MVS is designed to wrk with Archives and compression methods starting with the PKZIP 2.x standard. Although the implode algorithm was used in PKZIP 1.x, PKZIP for MVS 5.5 retains the ability to extract the older compression method's files.

Internal to the Zip Archive, file dates are saved as a count of the number of years from 1980. Because only six bits are used to store this date, a limit of 64 years (2**^) can be symbolized. This representation will successfully allow dates to be shown through the year 2043.

Environmental Execution Considerations

Region Size and Storage Usage

REGION space and 24/31-bit storage requirements - Older versions of PKZIP (v2.x) would always use work files to translate and then compress data before it was added to an archive file. Using these work files, very little REGION space was needed to run a job, since this space was used to handle the processing once the REGION had been consumed. Note that this situation can create a substantial amount of I/O.

PKZIP for MVS™ Version 5.5 increases the REGION requirement to 6M. The purpose behind this requirement is to increase speed and reduce I/O. However, if you run out of virtual storage then temporary files must be used to hold work space information. MEMORY_MODEL(MEDIUM or SMALL), will give PKZIP the outlet that it needs to handle the condition.

PKZIP processing, attempts to keep file management control information and compressed data in 31-bit virtual storage to maximize performance. In the event that 31-bit storage is constrained (by combinations of installation restrictions, high file volumes, and high data volumes), the following commands may be used to reduce 31-bit storage requirements for a given run.

- –DATA_STORAGE
- –MULTI_THREAD_LIMIT
- –MEMORY_MODEL(SMALL|MEDIUM|LARGE) controls where file management control blocks are held, such as, control blocks describing an archive file with its attributes.

When –MEMORY_MODEL(LARGE) is specified or defaulted, all PKZIP control blocks are held in 31-bit virtual storage.

When either SMALL or MEDIUM is specified, the file descriptor information is spilled to a set of work files to be sorted, merged, and selected. Note that file descriptors are built for both files existing in the input archive and new files to be selected, so the aggregate count must be managed. Approximate sizes for each file descriptor are as follows:

- VSAM - 2.5K.
- Sequential - 800 bytes.

- PDS/PDSE - 800 bytes for base dataset + 224 bytes per member.
- –DATA_STORAGE(MAX|xM) controls the amount of 31-bit virtual storage used to hold transient compressed data. When the amount of storage specified is exceeded, the data is processed through work files (controlled by the –TEMP_... suite of commands).
- –MULTI_THREAD_LIMIT(number) specifies the number of concurrent subtask sets to run for ZIP or UNZIP processing. When a count greater than 1 is used, additional copies of modules, work areas, and buffers are allocated to handle the processing.

Reserved DDNAMEs

The following DDNAMES are reserved for use by PKZIP for MVS:

ARCHTEMP - used for STAGE_TAPE_TO_DISK(y).

PKSPRINT - alternate SYSPRINT DD name when directed to a file.

ZPDIRIN - used when processing requires input Archive file descriptors to be spilled to work file.

ZPDIRSRT - used when processing requires input Archive file descriptors to be sorted in a work file.

ZPFILIN - used when input file descriptors requires sorting.

ZPFILSRT - used when input file descriptors require sorting.

ZIPCDS - license control dataset.

The following DDNAMES are reserved, but may be modified with a customized ACZDFLT module:

ARCHIN - –ARCHIVE_INFILE

ARCHOUT - –ARCHIVE_OUTFILE

PARMLIB - –DDNAME_PARMLIB

SYSIN - –DDNAME_SYSIN

SYSPRINT - –DDNAME_SYSPRINT

ZPSRTIN - –DDNAME_ZPSORTIN

ZPSRTOUT - –DDNAME_ZPSORTOUT

SYSPRINT

By default (unless overriden in the ACZDFLT module with –DDNAME_SYSPRINT, //SYSPRINT is used for PKZIP logging. This does not conflict with utilities used internally unless the SYSPRINT is directed to a physical file . Because utilities such as SORT may use a different set of DCB characteristics than PKZIP, a change to PKSPRINT for sysout will occur. See PKSPRINT.

PKSPRINT

//PKSPRINT is used when the SORT utility is internally invoked and the //SYSPRINT DD statement is determined to be allocated as a non-JES SYSOUT file. If not already allocated to the jobstep, PKZIP will dynamically allocate this DD to the SYSOUT= value specified in –SYSPRINT_SYSOUT_CLASS from the installation defaults module.

PKNODUMP

If allocated to the job step before invoking PKZIP, a //SYSABEND DD will not be dynamically allocated.

Use of System Utilities

SORT

PKZIP for MVS™ uses the system SORT utility to manage Archive Directory entries, during both match/merge procedures and View processing.

Access Method Services

PKZIP for MVS™ invokes this utility to locate cataloged files, define VSAM clusters, and handle Delete/Rename processing for an updated Archive.

IEBGENER

IEBGENER is called to open the PANVALET input stream (according to the –DDNAME_SYSIN specification in the active ACZDFLT module) and copy the data. The temporary file will be dynamically allocated with the –TEMP_SPACE_TYPE settings.

Chapter 3. - Installation, Licensing, and Configuration

Installation Overview

The installation of ***PKZIP for MVS™*** is accomplished by following the instructions as summarized below:

- Selecting the media to be used in installing PKZIP for MVS Version 5.5.
  - Installation from Downloaded File or CD.
  - Installation from Tape.
- Review the README.TXT file for recent information updates.
- Evaluate System Requirements.
- Edit the supplied job control (JCL) with appropriate parameter changes for your data center.
- Review the Installation, License, and Configuration chapter in this manual and proceed accordingly.
- Run the Installation Verification jobs and test product features by modifying the sample JCL supplied in PKZIP.MVS.INSTLIB.

Begin using the product (within any constraints posed by the licensing).

Details of these summarized instructions may be found below.

Type of Media Distribution for Installation

The ***PKZIP for MVS™*** may be received and installed from a variety of media types: :

- Downloaded from the PKZIP web sitehttp://www.pkzip.com/downloads
- Download from the PKWARE FTP site (ftp.asizip.com /pub/products/pkzip/mvs).
- Received from PKWARE on compact disc (CD).
- Received from PKWARE on magnetic cartridge (3480/3490).

Installation from Downloaded File or CD

If you have downloaded PKZIP for MVS from PKWARE's web site, ftp site, or received the product on CD, then the file you need to start with is the self extracting zip file called PKMVS.EXE. The PKMVS.EXE file contains the nine binary XMIT files needed for installation along with various other supporting text and documentation. The most relevant of the documentation is the README.TXT, as it will assist you with your installation and licensing of the product.

Some files extracted from PKMVS.EXE include:

PKZIP.XMIT.CEXEC	Compiled REXX Library
PKZIP.XMIT.HELP	Help Library
PKZIP.XMIT.INSTLIB	Install Library

PKZIP.XMIT.LOAD	Load Library
PKZIP.XMIT.MACLIB	Macro Library
PKZIP.XMIT.SPKZCLIB	REXX Exec Library
PKZIP.XMIT.SPKZMLIB	Message Library
PKZIP.XMIT.SPKZPLIB	Panel Library
PKZIP.XMIT.SPKZTLIB	Table Library
README.TXT	Installation and Configuration

It is necessary for you to review the installation instructions found within the README.TXT files if you are installing from download or CD. If the software was received as any other media type, a separate page of installation instructions will be enclosed in the packaging. In either case, follow the instructions applicable to your installation method before continuing through this document.

At this point, please refer to the respective README.TXT file that came with your download or CD. THIS ENDS THE INSTALLATION AND CONFIGURATION OF PKZIP IF YOU ARE INSTALLING FROM PKMVS.EXE. Resume your installation from the README.TXT file, if you are installing from a 3480 or a 3490 cartridge, then please continue onto the next section.

Installing from 3480 or 3490 Tape

If you have received PKZIP for MVS on a 3480 cartridge, the installation is as simple as an IEBCOPY of the PKZIP for MVS libraries from Tape to DASD.

The screen below shows the first step of the IEBCOPY. This represents one of nine steps needed to complete the installation of PKZIP for MVS 5.5 from Tape.

```
//JS010    EXEC PGM=IEBCOPY
//*
//SYSUT1   DD   DSN=PKWARE.MVS.CEXEC,
//              UNIT=tape,LABEL=(,SL),                    <===
//              DISP=OLD,VOL=(,RETAIN,,,SER=pkzip1)       <===
//*
//SYSUT2   DD   DSN=pkzip.mvs.CEXEC,                      <===
//              DISP=(NEW,CATLG,DELETE),
//              SPACE=(CYL,(2,1,5)),
//              UNIT=disk,                                <===
//              VOL=SER=volume                            <===
//*
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL,(5,5))              <===
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL,(5,5))              <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD   *
   COPY INDD=SYSUT1,OUTDD=SYSUT2
/*
```

If you do not prefer to type this entire job stream, you may download the COPYCART.TXT JCL from our web site at www.pkzip.com/downloads, upload it to a dataset or member. Remember to perform an ASCII or TEXT transfer to convert the data from ASCII to EBCDIC, modify the JCL, and submit.

The complete COPYCART JCL can also be found in APPENDIX D and in INSTLIB.

Initializing the License

Evaluation Period

License generation for a trial of the product allowing full use is a simple process of obtaining a key from the Sales Division. Once this process is completed PKZIP for MVS will allow access to all options for a period of 30 days. At some time during this process you must contact PKWARE to obtain licensing to allow use beyond the initial period. For Licensing, please contact the Sales Division at 937-847-2374 or email PKSALES@PKWARE.COM.

For Technical Support assistance, please contact the Product Services Division at 937-847-2687 or go to http://www.pkzip.com/support.

When you receive the license control card information from PKWARE you will build the license dataset using the build license program. There is a sample job stream in member LICUPDAT in the Installation Dataset (INSTLIB). By executing this job stream the LICENSE dataset will be updated and a report will be produced that will reflect the state of PKZIP for MVS at your location.

Show System Information

To display hardware and software information at your location, run the sample job stream in member LICSHSYS in the Installation Dataset (*pkzip.mvs*.INSTLIB). By executing this job stream a Show System Information report will be displayed.

Following is a sample of the report:

```
ZPLI210I PKZIP for MVS (TM), DATA COMPRESSION,
Copyright. 2003 PKWARE of Ohio, Inc.  All rights reserved.
PKZIP (R) IS A REGISTERED TRADEMARK OF PKWARE (R), INC.
For Licensing, please contact the Sales Division
at 937-847-2374 or email PKSALES@PKWARE.COM.
For Technical Support assistance, please contact the
Product Services Division at 937-847-2687 or go to
http://www.pkzip.com/support.

Friday 11/22/2002 (2002.326)  10:17:55
CPU model 2066 with 1 online
CPU serial number for CPU 0 is 01824A2066 (1824A), version code 00.
Service units per second per online CPU is 5612.07.
Approximate total MIPS (SUs/SEC / 48.5 * #CPUs) is 115.71.

Central Processing Complex (CPC) Node Descriptor:
  CPC ND = 002066.0B1.IBM.02.00000001824A
  CPC ID = 00
  Type(002066) Model(0B1) Manufacturer(IBM) Plant(02) Seq
Num(00000001824A)
JES2 z/OS 1.2 N1
DFSMS z/OS 1.3.0
```

Reporting the PKZIP for MVS 5.5 License

The procedures below will describe how to obtain this report.

- Edit the *.PKZIP.INSTLIB(LICPRINT) member, supply a job card, and substitute the following default line:

```
000400 //LICENSE PROC HLVL=PKZIP.MVS
```

"PKZIP.MVS" represents the high level qualifier for your installation.

Submit this job and the output should give you a return code of zero (RC=00) and the following additional lines.

```
ZPLI200I A LICENSE REPORT HAS BEEN REQUESTED ON 11/19/02 AT 8:35am IN
QZIP.FPD.LICENSE
ZPLI200I For Technical Support assistance, please contact Product Services
Division
ZPLI200I at 937-847-2687 or go to http://www.pkzip.com/support.
********************************************************************************
******************
ZPLI200I THIS PRODUCT IS LICENSED TO  CUSTOMER # 000012805
ZPLI200I  - CUSTOMER NAME -   PKWARE of Ohio, Inc.
ZPLI200I CPU model 2066 with 1 online
ZPLI200I CPU serial number for CPU 0 is 01824A2066 (1824A), version code 00.
ZPLI200I Service units per second per online CPU is 5612.07
ZPLI200I Approximate total MIPS (SUs/SEC / 48.5 * #CPUs) is 115.71
ZPLI200I Central Processing Complex (CPC) Node Descriptor: CPC ND =
002066.0B1.IBM.02.00000001824A
ZPLI200I   CPC ID = 00   Type(002066) Model(0B1) Manufacturer(IBM) Plant(02) Seq
Num(00000001824A)
********************************************************************************
******************
ZPLI200I COMPRESSION IS LICENSED ON THE FOLLOWING PROCESSORS
ZPLI200I SERIAL# *2824A PROCESSOR TYPE  2066 VERSION  00  WITH AN EXPIRATION DATE
OF  02/28/2400
ZPLI200I DECOMPRESSION IS LICENSED ON THE FOLLOWING PROCESSORS
ZPLI200I SERIAL# *2824A PROCESSOR TYPE  2066 VERSION  00  WITH AN EXPIRATION DATE
OF  02/28/2400
ZPLI200I SEQUENTIAL FILE HANDLER IS LICENSED ON THE FOLLOWING PROCESSORS
ZPLI200I SERIAL# *2824A PROCESSOR TYPE  2066 VERSION  00  WITH AN EXPIRATION DATE
OF  02/28/2400
```

Applying a License Key or Authorization Code

- Transfer the license file, provided by PKWARE, from the PC to the host. Be sure to convert the data from ASCII to EBCDIC and insert CR/LF's. Copying the authorization code from the text file and pasting it to the LICENSE member of the INSTLIB is an acceptable alternative.
- After the file has been transferred or copied to the host, edit the INSTLIB(LICUPDAT) member, supply a job card, and modify the following line of JCL:

```
000400 //LICENSE PROC HLVL=PKZIP.MVS,URUNIT=SYSDA,URVOL=WORK01
```

"PKZIP.MVS" is your high level qualifier for your installation. URUNIT and URVOL are the target unit and volume for the installed PKZIP product.

PKZIP for MVS Grace Period

PKWARE recognizes that there may be periods where the licensing environment established by the customer is no longer valid. Circumstances such as disaster recovery processing or the installation or upgrade of new processors will affect the environment. To accommodate the installation, PKZIP for MVS has a process that will allow you to continue to use the product for a period of 5 days. During this time, error messages will be displayed on the console (and the printout) for each execution of PKZIP for MVS. At the end of the grace period, if the license is not updated, the product will no longer function for the new CPU's in

any environment other than to VIEW an archive. This 5-day grace period is designed in such a way that it will not cease to function on a weekend or the Monday following the 5-day grace period. During this period you must contact PKWARE at pkcustomerservice@pkware.com to obtain licensing to allow use beyond the conditional period.

Running a Disaster Recovery Test

There are no special procedures necessary in order for you to use PKZIP during a Disaster Recovery Test. Because PKZIP licensing allows for such contingencies, the user can perform the following process to have PKZIP run at the DR site with a RC=00.

First, copy the production image of PKZIP from the production system over to the Disaster Recovery system.

Once on the system, simply run PKZIP from the CPU you want, and PKZIP will run conditionally for 5 days with a RC=00.

Again, it is important to contact PKWARE pkcustomerservice@pkware.com to resolve the licensing conflict within this time frame, if necessary.

Tailoring Site Specific Changes to the Defaults Module

The configuration defaults module, *.PKZIP.LOAD(**ACZDFLT**), is provided with the product. It is coded to allow for execution in a generic MVS environment. However, to make changes to the defaults, you will need to modify the *.PKZIP.INSTLIB(ACZDFLT) module. YOU MUST MODIFY THIS MODULE BEFORE YOU PROCEED TO USE PKZIP. It is recommended that the values defined in the module be reviewed before running in a production setting.

Upgrade note: Installations suppressing the //SYSIN PDS member verification for performance reasons with PROC_OPT1=N (available with 5.0.10 maintenance and above) in ACZDFLT should change to –CHECK_SYSIN_MEMBER=N in the assembly of ACZDFLT. PROC_OPT1 will no longer be used for this purpose in Release 5.5 and above.

```
MCZDFLTS TYPE=CSECT,                *
  LICENSE_HLQ=PKZIP.MVS,            * <== Change this to reflect your installation
  PARMLIB_DSNAME_ZIP=NULLFILE       *
  PARMLIB_DSNAME_UNZIP=NULLFILE,    *
```

Once you have, at minimum, modified the –LICENSE_HLQ statement to reflect your installation, you will need to assemble these changes via the **ASMDFLT** member in the *.PKZIP.INSTLIB to assist in creating a customized defaults module.

You may modify the other values in this module, or you may add to it. At minimum, the above three lines need to be modified or validated.

The table below represents the contents of the PKZIP for MVS defaults module. This table explains, in brief, the default parameters of the ACZDFLT's member and their relevance.

–LICENSE_HLQ	The high-level qualifiers of the xxx.LICENSE dataset. **–LICENSE_HLQ=** must be the same as the high-level qualifier used

	for the ***PKZIP for MVS™*** installation. The default qualifier is **PKZIP.MVS**. See also: $INSTLIC and LICxxxx members.
–ARCHIVE_UNIT **–OUTFILE_UNIT** **–TEMP_UNIT**	Device types to use during dynamic allocation request for non-VSAM files.
–ARCHIVE_STORCLASS **–OUTFILE_STORCLASS** **–TEMP_STORCLASS** **–VSAM_STORCLASS**	In DF/SMS environment, dynamic allocation information in lieu of volume allocation specifications.
–ARCHIVE_VOLUMES **–OUTFILE_VOLUMES** **–TEMP_VOLUMES** **–VSAM_VOLUMES**	Dynamic allocation target volumes for non-DF/SMS datasets. These are optional for non-VSAM datasets but are required for VSAM DEFINE CLUSTER control cards.

Protecting Files with the SAFETYEX Module

As delivered, the **SAFETYEX** module will protect PKUNZIP from overwriting **SYS1**. dataset names. If you would like to remove this restriction or add additional restrictions, you will need to edit the **SAFETYEX** source member in *.PKZIP.**INSTLIB**, make and save your changes, and run the **ASMSAFE** member of the *.PKZIP.INSTLIB to protect any files you specify from UNZIP overwrite processing.

If you do not want to make any changes to this module, then there is nothing that you need to do.

Activating the ISPF Interface

Activation of the PKZIP for MVS ISPF interface is accomplished as follows:

During product installation the PKZIP for MVS 5.5 ISPF libraries are loaded to disk. The high level qualifiers *(dsnhlq)* are selected by the user during the installation process.

To configure PKZIP ISPF, the user will need to make a few modifications to the PKZIP.INSTLIB(PKZSTART) member. Make the following changes to reflect your installation.

If the user environment can not support compiled REXX change the value of ’env’ to 'EXEC'. If your environment *does* support compiled REXX, then you do not have to change anything on this line. This defaults to 'CEXEC'.

Change the value of 'ispfhlq' to reflect the high level qualifier for your installation. This defaults to 'PKZIP.MVS'.

Change the value of 'llib' to indicate the name of the installed load library. The default is 'PKZIP.MVS.LOAD'.

env = 'CEXEC'

ispfhlq = 'PKZIP.MVS'

llib = 'PKZIP.MVS.LOAD'

Now save your changes to the PKZSTART member.

To quickly test whether the user configuration has worked, simply type "EXEC" next to the PKZSTART member. If everything has gone accordingly during the installation, after typing in “EXEC”. the user should be prompted to enter the configuration screen for PKZIP for MVS.

You may choose to add the PKZSTART member to a REXX exec in your SYSEXEC or SYSPROC concatenation that will initialize the ISPF interface. If the user prefers to activate the PKZIP for MVS ISPF from your ISPF main menu, add an entry that will activate PKZIP for MVS. Both methods are explained in the following paragraphs. Significant performance improvements can be achieved by using the compiled REXX exec.

ISPF Main Menu

To execute PKZIP from an ISPF menu panel you must add an entry to the main menu for ISPF. This is normally a panel named (ISR@PRIM). Add the following line (or whatever the user deams appropriate) to the BODY section of the panel definition:

```
P  PKZIP for MVS 5.5 ISPF
```

Add the following line to the PROC section:

```
P,'CMD(%PKZSTART)'
```

Replace the ‘**P**’ with whatever main menu option you added in the BODY section of the panel definition. The user will notice that the PKZSTART exec has an argument passed to it. The argument ‘CEXEC’ causes the libraries containing the compiled REXX routines to be allocated. The user will gain significant increases in performance by using these libraries. If your operating system release or any other reason might prevent you from using the compiled REXX, then call PKZSTART with the argument of ‘EXEC’ and the normal interpreted REXX libraries will be used.

PKZSTART is the initial exec that starts the interface and it also allocates the necessary ISPF application libraries. Consequently, it must be modified to reflect the installed library names (as it was documented in the previous section).

Chapter 4. - Getting Started with *PKZIP for MVS*™

PKZIP for MVS™ is a broad, flexible product on the MVS/ESA, OS/390, and z/OS platforms, allowing for compression/decompression and encryption of files. It is fully compliant with other ***PKZIP***® 2.x compatible compression products running on other operating systems. However, if you are licensed for PKZIP for MVS 5.5 Advanced Encryption, this feature is only compatible with other PKWARE products enabled for this feature.

Because the ***PKZIP***® standard for text data storage is ASCII, ***PKZIP for MVS***™ facilitates conversion between the ASCII and EBCDIC character sets. Therefore, compressed text files can be transferred between IBM mainframe environments and systems using either character set. Some of these platforms include DOS, Windows, UNIX, OS/400, and PKZIP.

In addition to ***PKZIP***® archive format support, ***PKZIP for MVS***™ can also produce and manipulate (GNU) GZIP-format archives. Additional information on this subject can be found in Chapter 10. - Processing with GZIP.

Introduction to PKZIP for MVS

PKZIP for MVS consists of two separate executable programs:

1. PKZIP - providing compression of datasets into an archive.
2. PKUNZIP - providing decompression of datasets from an archive.

To use either of these programs, you must specify:

- Commands, which tell PKZIP or PKUNZIP what processing they are to perform and how they are to do it. Commands are identified by a preceding hyphen ("**–**"). For example, **–**ARCHIVE_DSN is the command that designates the dataset name for the ZIP Archive containing compressed data.
- File selections, which identify the files to be compressed into an archive (PKZIP) or decompressed from an archive (PKUNZIP). File selections are distinguished from commands because they are not preceded by a hyphen ("**–**").
- These commands and file selections can be specified in a number of ways. The most common way, which is the way that will be used in the examples presented in this chapter, is to run PKZIP and PKUNZIP as batch jobs using JCL and specifying the commands and file selections via SYSIN, as shown in the next section.

Invoking PKZIP or PKUNZIP Using JCL

In these examples you will be running PKZIP for MVS in batch by submitting JCL. However, PKZIP for MVS can also be executed using the ISPF panels interface, called from a user written program, or from a TSO environment with REXX or CLISTS.

The example below demonstrates the basic JCL statements required to run PKZIP.

```
//<job card>
//ZIP      EXEC PGM=PKZIP[2],REGION=8M
//STEPLIB[4]  DD DISP=SHR,DSN=PKZIP.MVS.V55.LOAD
//SYSPRINT[5] DD SYSOUT=*
//SYSIN    DD *
-ARCHIVE_DSN(MY.ARCHIVE.ZIP)
<commands>[7]
/*
//
```

Notes for Invoking PKZIP or PKUNZIP Using JCL

1. ***<job card>*** should be replaced with the job details required for running this job, in accordance with your installation standards.

2. To add, update, freshen, delete, or view compressed files within a ZIP archive use the 'PKZIP' program. To extract, test, or view compressed files in a ZIP archive use the 'PKUNZIP' program.

3. PKZIP for MVS should normally run within a region size of 8Mb, however this value is dependent on the number and type of files being processed. If you encounter storage problems then this value should be increased if possible.

4. STEPLIB specifies the library that contains PKZIP for MVS. PKZIP may be placed in the JOBLIB DD or in one of the libraries shared by all MVS processing, for example, LNKLST, in which case there is no need to use the STEPLIB DD.

5. SYSPRINT contains all the message output by PKZIP. A SYSABEND DD card will be dynamically allocated by default if one is not supplied.

6. SYSIN is the usual mechanism for supplying commands to PKZIP. Alternatively you can use the PARM parameter on the EXEC statement, the //PARMLIB DD, or a combination of all three.

7. Commands specify the processing to be carried out by PKZIP.

Return Codes

PKZIP issues a completion code dependent on the results of the processing that was carried out. The completion code can take the following values:

0	Processing has completed without errors being detected.
4	A warning message has been output but processing has continued.
8 or higher	An error has occurred during processing; refer to the error messages for more details.

The final completion code issued is the maximum value of the conditions found during the sum. A return code greater than zero indicates that there are one or more warning or error messages in the job output.

Compressing a Dataset

The following example shows how to compress a dataset using PKZIP for MVS.

```
//ZIP      EXEC PGM=PKZIP,REGION=8M
//STEPLIB  DD DISP=SHR,DSN=PKZIP.MVS.V55.LOAD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
-ARCHIVE_DSNAME(MY.ARCHIVE.FILE.ZIP)
-ARCHIVE_UNIT(SYSDA)
MY.INPUT.DATA.SEQ
/*
```

This step will give the following output:

```
ZPLI001I PKZIP for MVS (TM), Data Compression, Version 5.5  -   09/06/02 13.16
ZPLI001I  Copyright.  2003 PKWARE of Ohio, Inc.  All Rights Reserved.
ZPLI001I  PKZIP (R) is a registered trademark of PKWARE (R), Inc.
ZPLI001I  Registered, Processor Type=7060 Processor Group=00 Serial Number=00052
ZPLI001I  OS Level: HBB7703  SP6.1.0
-ARCHIVE_DSN(MY.ARCHIVE.FILE.ZIP)
-ARCHIVE_UNIT(SYSDA)
MY.INPUT.DATA.SEQ
ZPAM030I OUTPUT Archive opened:  MY.ARCHIVE.FILE.ZIP
ZPAM253I ADDED     File MY.INPUT.DATA.SEQ
ZPAM254I   as MY/INPUT/DATA/SEQ
ZPAM255I   (DEFLATED 93%/93%)
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

In this case, the sequential dataset MY.INPUT.DATA.SEQ is to be compressed into the new ZIP archive MY.ARCHIVE.FILE.ZIP, which is created on a SYSDA volume.

Notes for Dataset Compression

1. ZIP archives can be considered as a large envelope or box into which the compressed files are placed. Note however, that an empty dataset is not the same as an empty archive. ZIP archives created by PKZIP for MVS cannot be pre-allocated, only PKZIP should be used to create new archives.

2. You tell PKZIP how to create the ZIP archive. By default ZIP archives are created as sequential datasets and allocated using half track blocking. But, if required, you have full control over the type of archive created and how they are created using the various –ARCHIVE_* commands available.

3. PKZIP will compress datasets using a ‘file selection’. Any command that does not begin with a “-” is considered to be a file selection. In the previous example, we told PKZIP to compress the sequential dataset MY.INPUT.DATA.SEQ.

4. You can specify a file for compression via a INFILE_DD statement if you prefer, but a file selection has the advantage of wildcards. For example, to compress a specific group of files, you could type MY.INPUT.DATA.*. This file selection would inform PKZIP to compress every dataset that begins with the previous qualifying nodes PKZIP can compress up to 65, 535 datasets or up to 4Gb of data.

5. To ensure cross platform compatibility, all MVS dataset names are converted to the standard PKZIP UNIX format, such as, MY/INPUT/DATA/SEQ. When you unzip the file, the conversion is reversed to recreate the original MVS name. See –NIASEP for more information about the character used to separate levels.

The compressed version of the sequential dataset is known as a ZIPPED or compressed file within the ZIP archive.

Viewing the Contents of an Archive

The following example shows how to view the contents of a ZIP archive, created in the previous example, using PKZIP for MVS.

```
//STPZIP   EXEC PGM=PKZIP
//STEPLIB  DD DISP=SHR,DSN=PKZIP.MVS.V55.LOAD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
-ARCHIVE_DSN(MY.ARCHIVE.FILE.ZIP)
-ACTION(VIEW)
/*
```

This step will yield output similar to the following:

```
ZPLI001I PKZIP for MVS (TM), Data Compression, Version 5.5  -   09/06/02 13.16
ZPLI001I  Copyright.  2003 PKWARE of Ohio, Inc.  All Rights Reserved.
ZPLI001I  PKZIP (R) is a registered trademark of PKWARE (R), Inc.
ZPLI001I  Registered, Processor Type=7060 Processor Group=00 Serial Number=00052
ZPLI001I  OS Level: HBB7703  SP6.1.0
-ARCHIVE_DSN(MY.ARCHIVE.FILE.ZIP)
-ACTION(VIEW)
ZPAM030I INPUT  Archive opened:  MY.ARCHIVE.FILE.ZIP
ZPAM014I There are        1 file(s) in the input Archive.
ZPAM012I ZIP comment: PKZIP for MVS by PKWARE, Inc.
ZPAM013I ************************************************************************
ZPAM015I Length Method    Size Ratio   Date    Time   CRC-32   Name
ZPAM016I ------------ ------------ ------------ ----- ----------
ZPAM017I 1,067  Dflt-Norm 81   92%  09/06/2002 11:54 C7A3091B  MY/INPUT/DATA/SEQ
ZPAM016I ------------ ------------
ZPAM019I 1,067            81   92%
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Notes for Viewing the Contents of an Archive

1. The –ACTION(VIEW) command is available via the programs PKZIP or PKUNZIP.

2. The –ACTION(VIEW) command has various options that can be used to tailor the output. For example, if the user has multiple files within the archive then the output can be sorted by the file's attributes including name, size, and compression ratio.

3. This example demonstrates a standard view of the archive. It displays information about the files in the archive including the original length of the file, the compression method, and the compressed file size.

–ACTION(VIEWDETAIL)

One especially useful option is the –ACTION(VIEWDETAIL) control card. It displays the full technical details, including any file attributes stored, for each file in the archive.

```
//STPZIP   EXEC PGM=PKZIP
//STEPLIB  DD DISP=SHR,DSN=PKZIP.MVS.V55.LOAD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
-ARCHIVE_DSN(MY.ARCHIVE.FILE.ZIP)
-ACTION(VIEWDETAIL)
/*
```

This step could give the following output:

```
ZPLI001I PKZIP for MVS (TM), Data Compression, Version 5.5  -   09/06/02 13.16
ZPLI001I  PKZIP (R) is a registered trademark of PKWARE (R), Inc.
ZPLI001I  Registered, Processor Type=7060 Processor Group=00 Serial Number=00052
ZPLI001I  OS Level: HBB7703  SP6.1.0
-ARCHIVE_DSN(MY.ARCHIVE.FILE.ZIP)
-ACTION(VIEWDETAIL)
ZPAM030I INPUT  Archive opened:  MY.ARCHIVE.FILE.ZIP
ZPAM014I There are        1 file(s) in the input Archive.
ZPAM012I ZIP comment: PKZIP for MVS by PKWARE of Ohio, Inc.
ZPAM013I ****************************************
ZPAM001I Filename: MY/INPUT/DATA/SEQ
ZPAM002I File type:          TEXT
ZPAM003I Date/Time:          06-SEP-2002 11:54:06
ZPAM004I Compression Method: Deflate- Normal
ZPAM005I Compressed Size:    81
ZPAM006I Uncompressed Size:  1,067
ZPAM007I 32-bit CRC:         C7A3091B
ZPAM008I Created by:         PKZIP for MVS   5.5 * - 2.x compatible
ZPAM009I Needed to extract:  PKUNZIP     2.0
ZPAM301I File Type:          NONVSAM  SEQUENTIAL
ZPAM303I File Record Format: FB
ZPAM304I File Allocation Type: TRK
ZPAM305I File Primary Space Allocated: 1
ZPAM306I File Secondary Space Allocated: 1
ZPAM307I File Record Size: 80
ZPAM308I File Block Size: 3120
ZPAM309I File Volume(s) Used: SUP001
ZPAM310I File Creation Date: 2002/08/02
ZPAM311I File Referenced Date: 2002/09/05
ZPAM000I SMS Storage Class:      SUPPORT
ZPAM013I ****************************************************
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Note: The order in which attributes are displayed may vary.

Decompressing a Dataset

The following example shows how to extract or, "UNZIP", a dataset using PKZIP for MVS.

```
//UNZIP    EXEC PGM=PKUNZIP,REGION=8M
//STEPLIB  DD DISP=SHR,DSN=PKZIP.MVS.V55.LOAD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
-ARCHIVE_DSN(MY.ARCHIVE.FILE.ZIP)
-OUTFILE_UNIT(SYSDA)
/*
```

This step will give the following output:

```
ZPLI001I PKZIP for MVS (TM), Data Compression, Version 5.5  -   09/06/02 13.16
ZPLI001I  Copyright.  2003 PKWARE of Ohio, Inc.  All Rights Reserved.
ZPLI001I  PKZIP (R) is a registered trademark of PKWARE (R), Inc.
ZPLI001I  Registered, Processor Type=7060 Processor Group=00 Serial Number=00052
ZPLI001I  OS Level: HBB7703  SP6.1.0
-ARCHIVE_DSN(MY.ARCHIVE.FILE.ZIP)
-OUTFILE_UNIT(SYSDA)
ZPAM030I INPUT  Archive opened:  MY.ARCHIVE.FILE.ZIP
ZPEX002I MY/INPUT/DATA/SEQ
ZPEX003I Extracted to MY.INPUT.DATA.SEQ
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Notes for Decompressing a Dataset

1. To extract files from an archive you must call the PKUNZIP program.

2. The extracted dataset is created dynamically according to the stored file attributes, if any, or the –OUTFILE DD attributes supplied in the job allocation. In this case the dataset was recreated on a SYSDA volume. Information required to create the dataset that is not provided by the stored file attributes or by the –OUTFILE allocation may be defaulted by PKUNZIP.

3. By default, PKUNZIP will try to extract every file that is compressed and stored inside the ZIP file or archive. To extract just one file or several files, you will need to explicitly state the files you wish to extract or decompress via file selection. Wildcards can be used in the file selection to have PKUNZIP extract a suite of like datasets.

4. If the extracted dataset already exists, then (by default) PKUNZIP will not overwrite it.

5. To overwrite a dataset or PDS member use the –OUTFILE_OVERWRITE command. To add new members to existing PDS's use the –INSERT_MEMBER command. Alternatively the extracted file can be given a new name if required by using the –UNZIPPED_DSN command.

Updating or Refreshing a File

You cannot –ACTION(ADD) a file that already exists in a ZIP archive, however you can replace it using the–ACTION(UPDATE) or –ACTION(FRESHEN) commands.

The –ACTION(UPDATE) and –ACTION(FRESHEN) commands differ in their processing of files that do not already exist in the archive. As a distinction, if a file selected for compression does not already exist in the archive then –ACTION(UPDATE) will add it, but -ACTION(FRESHEN) will ignore it.

Invoking *PKZIP for MVS*™ Services

There are several ways to use ***PKZIP for MVS***™ within the and MVS/ESA, OS/390, and z/OS operating environments. These include:

- Batch JCL job-steps.
- Started Task JCL.
- Executed from TSO CLIST/REXX.
- TSO Command Line Interface.
- ISPF Panel.
- API (Application Program Interface).

The following sections provide a brief overview of these interfaces. Subsequent sections in this chapter describe basic functions using the JCL interface.

Invoking the PKZIP or PKUNZIP Programs From JCL (Batch or Started Task)

PKZIP for MVS™ programs can be executed from a batch job or STC. See pkzip.mvs.INSTLIB(IVPBASIC) for a sample JOB, or use the ISPF interface to generate JCL for a Batch job.

Invoking the PKZIP or PKUNZIP Programs as Called Programs Under TSO

The ***PKZIP for MVS***™ batch interface programs can be executed within a TSO CLIST or REXX EXEC provided that the proper FILE allocations (TSO equivalent of DD statements) are made.

The following samples show how allocations can be done to invoke ***PKZIP for MVS***™.

CLIST Call - Read commands from a member and put messages to a pre-allocated FB132 file.

```
PROC 0
ALLOC F(SYSIN) DA('pkzip.mvs.INSTLIB(SAMPVIEW)') SHR
ALLOC F(SYSPRINT) DA('USERID.QZ.SYSOUT') SHR
CALL 'pkzip.mvs.LOAD(PKUNZIP)'
FREE F(SYSIN,SYSPRINT)
```

REXX Call - Pass commands as a parm and allocate a new SYSPRINT file to browse.

```
/*  Rexx    Sample call of PKUNZIP for -VIEW with no SYSIN  */

/* First allocate a SYSPRINT output file for later browsing  */

Address TSO
"attrib dcbout recfm(f b) lrecl(132) blksize(27984)"
"ALLOC F(SYSPRINT) da(my.sysprint) new catalog cylinders " ,
       "using(dcbout) space(1,1)"
```

```
/* Define the command list to pass (without SYSIN)           */

callparms = "-NOSYSIN -ARCHIVE(USERID.MY.ZIP) -VIEWBRIEF"

/* Invoke PKUNZIP                                             */

Address LINKMVS "PKUNZIP callparms"

/* Free the work files and browse the output                  */

Address TSO
"free f(DCBOUT,SYSPRINT)"
Address ISPEXEC
"browse dataset(my.sysprint)"
```

Invoking ZIP or UNZIP TSO Command Line Interface

A subset of ***PKZIP for MVS***™ features can be invoked from the ZIP and UNZIP REXX EXECs. These commands are intended to approximate the PKZIP and PKUNZIP DOS-based commands with similar command syntax. In addition to the standard commands being passed as input options, several shorthand Actions and Options are provided with this interface (see the tables below).

Syntax

```
ZIP <-action> [-options] <Archive_name> <File_names>
```

```
UNZIP <-action> [-options] <Archive_name> <File_names>
```

Valid ZIP Actions

'–a'	'–ACTION(ADD)'
'–d'	'–ACTION(DELETE)'
'–f'	'–ACTION(FRESHEN)'
'–u'	'–ACTION(UPDATE)'
'–v'	'–ACTION(VIEW)'
'–vbd'	'–ACTION(VIEWDATE)'
'–vn'	'–ACTION(VIEWNAME)'
'–vo'	'–ACTION(VIEWOFFSET)'
'–vp'	'–ACTION(VIEWPERCENT)'
'–vs'	'–ACTION(VIEWSIZE)'
'–vr'	'–ACTION(VIEWREVERSE)'
'–vrd'	'–ACTION(VIEWDATEREVERSE)'
'–vrn'	'–ACTION(VIEWNAMEREVERSE)'
'–vro'	'–ACTION(VIEWOFFSETREVERSE)'
'–vrp'	'–ACTION(VIEWPERCENTREVERSE)'
'–vrs'	'–ACTION(VIEWSIZEREVERSE)'
'–vb'	'–ACTION(VIEWBRIEF)'
'–vbd'	'–ACTION(VIEWBRIEFDATE)'
'–vbn'	'–ACTION(VIEWBRIEFNAME)'
'–vbo'	'–ACTION(VIEWBRIEFOFFSET)'
'–vbp'	'–ACTION(VIEWBRIEFPERCENT)'
'–vbs'	'–ACTION(VIEWBRIEFSIZE)'
'–vbr'	'–ACTION(VIEWBRIEFREVERSE)'
'–vbrd'	'–ACTION(VIEWBRIEFDATEREVERSE)'
'–vbrn'	'–ACTION(VIEWBRIEFNAMEREVERSE)'

'–vbro'	'–ACTION(VIEWBRIEFOFFSETREVERSE)'
'–vbrp'	'–ACTION(VIEWBRIEFPERCENTREVERSE)'
'–vbrs'	'–ACTION(VIEWBRIEFSIZEREVERSE)'
'–vt'	'–ACTION(VIEWDETAIL)'
'–vtd'	'–ACTION(VIEWDETAILDATE)'

Valid ZIP Options

'–vtn'	'–ACTION(VIEWDETAILNAME)'
'–vto'	'–ACTION(VIEWDETAILOFFSET)'
'–vtp'	'–ACTION(VIEWDETAILPERCENT)'
'–vts'	'–ACTION(VIEWDETAILSIZE)'
'–vtr'	'–ACTION(VIEWDETAILREVERSE)'
'–vtrd'	'–ACTION(VIEWDETAILDATEREVERSE)'
'–vtrn'	'–ACTION(VIEWDETAILNAMEREVERSE)'
'–vtro'	'–ACTION(VIEWDETAILOFFSETREVERSE)'
'–vtrp'	'–ACTION(VIEWDETAILPERCENTREVERSE)'
'–vtrs'	'–ACTION(VIEWDETAILSIZEREVERSE)'
'–ex'	'–COMPRESSION_LEVEL(MAXIMUM)'
'–en'	'–COMPRESSION_LEVEL(NORMAL)'
'–ef'	'–COMPRESSION_LEVEL(FAST)'
'–es'	'–COMPRESSION_LEVEL(SUPERFAST)'
'–e0'	'–COMPRESSION_LEVEL(STORE)'
'–s…'	secure with encryption where "…"=password
'–noprompt'	When being run from an ISPF environment, the default is for the interpreted commands to be displayed in an EDIT session allowing you an opportunity to alter the commands. This option will bypass this feature, as well as, the ISPF browse of SYSPRINT when the function is complete.

Valid UNZIP Actions

'–e'	'–ACTION(EXTRACT)'
'–o'	'–OUTFILE_OVERWRITE(Y)'
'–v'	'–ACTION(VIEW)'
'–t'	'–ACTION(TEST)'
'–vbd'	'–ACTION(VIEWDATE)'
'–vn'	'–ACTION(VIEWNAME)'
'–vo'	'–ACTION(VIEWOFFSET)'
'–vp'	'–ACTION(VIEWPERCENT)'
'–vs'	'–ACTION(VIEWSIZE)'
'–vr'	'–ACTION(VIEWREVERSE)'
'–vrd'	'–ACTION(VIEWDATEREVERSE)'
'–vrn'	'–ACTION(VIEWNAMEREVERSE)'
'–vro'	'–ACTION(VIEWOFFSETREVERSE)'
'–vrp'	'–ACTION(VIEWPERCENTREVERSE)'
'–vrs'	'–ACTION(VIEWSIZEREVERSE)'
'–vb'	'–ACTION(VIEWBRIEF)'
'–vbd'	'–ACTION(VIEWBRIEFDATE)'
'–vbn'	'–ACTION(VIEWBRIEFNAME)'
'–vbo'	'–ACTION(VIEWBRIEFOFFSET)'
'–vbp'	'–ACTION(VIEWBRIEFPERCENT)'

'–vbs'	'–ACTION(VIEWBRIEFSIZE)'
'–vbr'	'–ACTION(VIEWBRIEFREVERSE)'
'–vbrd'	'–ACTION(VIEWBRIEFDATEREVERSE)'
'–vbrn'	'–ACTION(VIEWBRIEFNAMEREVERSE)'
'–vbro'	'–ACTION(VIEWBRIEFOFFSETREVERSE)'
'–vbrp'	'–ACTION(VIEWBRIEFPERCENTREVERSE)'
'–vbrs'	'–ACTION(VIEWBRIEFSIZEREVERSE)'
'–vt'	'–ACTION(VIEWDETAIL)'
'–vtd'	'–ACTION(VIEWDETAILDATE)'
'–vtn'	'–ACTION(VIEWDETAILNAME)'
'–vto'	'–ACTION(VIEWDETAILOFFSET)'
'–vtp'	'–ACTION(VIEWDETAILPERCENT)'
'–vts'	'–ACTION(VIEWDETAILSIZE)'
'–vtr'	'–ACTION(VIEWDETAILREVERSE)'
'–vtrd'	'–ACTION(VIEWDETAILDATEREVERSE)'
'–vtrn'	'–ACTION(VIEWDETAILNAMEREVERSE)'
'–vtro'	'–ACTION(VIEWDETAILOFFSETREVERSE)'
'–vtrp'	'–ACTION(VIEWDETAILPERCENTREVERSE)'
'–vtrs'	'–ACTION(VIEWDETAILSIZEREVERSE)'

To compress and store all of a user's files into their own archive, type the following:

ZIP –a 'MY.CLI.TEST.ZIP' '&SYSUID.**'

Invoking the PKZIP for MVS™ ISPF Panel Interface

The ISPF panel interface provides a simple way for a TSO user to either build batch JCL or invoke foreground **PKZIP for MVS™** services. The panel interface also provides a dynamic table interface to display ZIPPED files within a ZIP Archive allowing line-command selection for browsing, viewing, and extracting.

```
                          PKZIP for MVS™ 5.5 Main Menu

OPTION  ===>

   C  Config           Display/Modify configuration
  ZD  Zip Defaults     Display default settings for ZIP processing
  UD  Unzip Defaults   Display default settings for UNZIP processing
   U  Unzip            Decompress File(s) Stored in Archive File
   V  View             Display the contents of an Archive File
   Z  Zip              Compress File(s)and Write to an Archive File

   S  Sysprint         Browse Log of last on-line execution
   M  Messages         Message ID lookup
   L  License          Display License Information

W  What's New          Browse Information on changes since last release
A  Contact PKWARE      Browse Information on how to contact PKWARE

   X  EXIT             Exit PKZIP for MVS™


   To EXIT Press PF3 or enter X       For HELP Press PF1
```

This topic is covered in detail in Chapter 11. - Using the ISPF Interface. Detailed instructions for installation and implementation can be found in Chapter 3. - Installation, Licensing, and Configuration.

Invoking the PKZIP for MVS™ ZIP and UNZIP Services API

The ***PKZIP for MVS™*** Callable Services API provides an interface to the compression and decompression algorithms. This allows MVS, OS/390, and z/OS applications to compress and decompress storage, memory, data, records, and buffers dynamically without using ZIP Archive files. More details regarding the ZIP API are in Chapter 12. This interface is not distributed with the basic product. To use the ZIP API, you will need to contact PKWARE to make the appropriate arrangements. Call (937) 847-2374 to speak to a Technical Support representative.

Configuration Manager

General Purpose

In releases of PKZIP for MVS prior to 5.0, users were allowed to create a configuration file that allowed PKZIP to accept different parms during a run of PKZIP or PKUNZIP. ***PKZIP for MVS™*** 5.x has implemented an extended means of allowing the user to control the defaults that PKZIP/UNZIP uses during a job.

The defaults for PKZIP is set by first editing PKZIP.MVS. INSTLIB(ACZDFLT). These defaults are then assembled into PKZIP.MVS. LOAD by using the ASMDFLT member of INSTLIB. The ACZDFLT's module gives you extended flexibility to make PKZIP work the way you want it to - not the way the vendor provides it.

ACZDFLT is a data-only CSECT which uses macro MCZDFLTS to generate the table data. An installation can customize the values for this module by adding appropriate variable data to the invocation of MCZDFLTS in the ACZDFLT module source.

Multiple versions of ACZDFLT may be assembled and linked into an execution load library for use with the –DM execution parameter. This allows multiple configurations to be pre-defined and used. In addition to the //PARMLIB DD for the Configuration File, //CONFIG DD is also supported for compatibility with ***PKZIP for MVS™*** 2.61 and below.

Making Changes to the Defaults

Within the ACZDFLT's member, there is a minimum of one variable that needs to coincide with your installation's PKZIP high level qualifier. This variable is the LICENSE_HLQ parameter. PKZIP accesses your PKZIP.MVS. LICENSE dataset during every execution of ZIP or UNZIP, and providing your installation's high level qualifier for the LICENSE_HLQ parameter tells PKZIP where to find it.

```
*********************************************************
MCZDFLTS TYPE=CSECT,                                    *
      LICENSE_HLQ=PKZIP.MVS                             *
*********************************************************
```

Remember that the PKZIP.MVS. INSTLIB(ACZDFLT) is a configuration member. Therefore, besides providing the high level qualifier for your installation, you can re-establish new defaults for PKZIP and PKUNZIP processing. Below is an example of other parameters that can be coded.

```
************************************************************
MCZDFLTS TYPE=CSECT,                                       *
      LICENSE_HLQ=PKZIP.MVS                                *
      PARMLIB_DSNAME_ZIP=NULLFILE,                         *
      PARMLIB_DSNAME_UNZIP=NULLFILE,                       *
      ARCHIVE_UNIT=SYSDA,                                  *
      TEMP_UNIT=SYSDA,                                     *
      COMPRESSION_LEVEL=SUPERFAST,                         *
      CRLF=C                                               *
***************** Bottom of Data **************************
```

Assembling Your Changes

After editing the ASMDFLT member of PKZIP.MVS.INSTLIB,modify the ASMDFLT JCL member per your JCL Standards and submit the job to assemble PKZIP.MVS.INSTLIB(ACZDFLT) into PKZIP.MVS. LOAD. For every execution of ZIP and UNZIP, PKZIP will refer to this assembled ACZDFLT module in your LOAD library.

Inputs

User inputs to ***PKZIP for MVS™*** can come from various sources and formats, as described in the following tables:

User Input Sources (MVS)

ACZDFLT or other customized defaults modules.	The installation defaults module, which is provided at installation time, or modified and re-assembled by the systems programmer responsible for installation changes.
Installation Configuration File	A list of commands can be defined in a sequential file (or PDS member). This file can either be dynamically allocated (file name defined in ACZDFLT), or explicitly allocated through the //PARMLIB DD statement.
//SYSIN DD	A batch, started-task or TSO user may provide this DD statement to input control statements.
EXEC PGM … PARM=	A batch job or started task can pass a subset of parameters through the execution PARM= statement.
API Call Parm	When calling ***PKZIP for MVS™*** from an application program, this set of parameters acts like EXEC PARM= above.

Processing Order of Control Statements

In general, after the loading of the defaults module ACZDFLT, control statements are read sequentially from the various sources in the order below.

1. Configuration File (//PARMLIB DD or dynamically allocated).
2. EXEC PARM, or API Call Parm.
3. //SYSIN DD.

Exceptions to this order are for commands providing early initialization control through the EXEC PARM.

–DM *ACZDFLT* <= **D**efaults **M**odule selection.

–ECHO.

Configuration Manager Processing: Managing Control Statements

Control Statement Definitions

Control statements are managed via an internal control table; ACMTABLE. This table determines which command values are permitted for each command and provides validation information to the Configuration Manager.

> Keywords, formats, and values generated in the defaults module are kept in synchronization with internal module control information maintained in ACMTABLE (which is used programmatically by Configuration Manager routines to parse control statements). The control statement values are mapped directly to the defaults module values for use.
>
> Default values for the commands are held in module ACZDFLT, which is loaded at run-time. A sample source module is provided (pkzip_product.INSTLIB(ASMDFLT)) can be assembled to change the defaults for the installation.
>
> In addition, ACZDFLT can be assembled as a different load module name to create custom profiles of defaults for a variety of needs. A different flavour of ACZDFLT can be requested at execution time by using the JCL EXEC parameter " –DM nnnnnnnn", where nnnnnnnn is the name of the module to use instead of ACZDFLT.

The ISPF interface has 2 options UD and ZD that allow you to see and set values for many of the commands. This may be used as a reference when trying to determine which of the available command values are to be used.

The batch –SHOW_SETTINGS command may aslo be helpful as a reference to command names and their default values.

Troubleshooting

PKZIP for MVS™ Messages

PKZIP for MVS™ writes messages to SYSPRINT (or other output DD file as specified by the defaults module) that indicate whether processing is successful. Each message type is defined with a unique message ID starting with "ZP" (see ***PKZIP for MVS™* Messages and Codes Guide** for specific format information).

The volume of messages that are written to SYSPRINT are controlled by the command –LOGGING_LEVEL. Additional processing information will be displayed when "VERBOSE" is requested. This does not affect the output of critical error messages, which will be written regardless of the level requested.

Explanatory information regarding messages can also be found on-line via the ISPF interface, or by browsing the PKZIP.MVS.HELP members.

When questions regarding which processing options are in effect, –SHOW_SETTINGS should be coded as the last SYSIN command or EXE PARM to display all final parameter values.

When isses concerning non-VSAM dataset allocation arise, –TRACE_DYNALLOC(3) should be specified to see values used for individual files.

When issues concerning VSAM Cluster definitions arise, –TRACE_AMS(1) should be used to see control cards passed to IDCAMS.

Chapter 5. File Selection and Name Processing

ZIP Processing File Selection

This section discusses how file selection is performed for ZIP processing with ***PKZIP for MVS™***. The primary commands that are used are discussed here, along with overview notes and known restrictions. It is important to keep in mind that ZIP file directory entries within a ZIP archive are defined in a system-independent format. The file directory entries within a ZIP archive are actually in a format compatible with UNIX systems and have been translated into the ASCII character set. In addition, the dataset level separators are typically set as the forward slash ("/"), not the period (".") as in MVS (although this can be controlled through command actions in ***PKZIP for MVS™***).

Primary File Selection Inputs

One of the initial ZIP processing phases involves selecting files to be compressed. Candidate files are identified as input parameters are processed and as the old archive directory is read (if one is provided). As a result, dataset selection is controlled by a combination of 3 input sources:

Selection Source	Effective –ACTION Processes
Cataloged Dataset name command requests.	ADD, UPDATE
`-INFILE` command (JCL DD) requests.	ADD, UPDATE
Input ZIP Archive files.	UPDATE, FRESHEN

Dataset names found with the above inputs are combined into a single list of candidate files to be processed in the compression phase. A dataset will be selected only once. The following sections describe aspects of file selection from each of the input sources.

Cataloged Dataset Name Filter Requests

When requesting a file (or set of files) for ZIP processing by dataset name, the result is a standard search of the system catalog structure to determine eligible file names. Both NONVSAM and VSAM CLUSTER entries will be used to identify candidates.

When dataset name masking is requested, multiple dataset names may be identified from the system catalog.

Also see: –RECURSE_LEVELS and –VSAM.

Exclusion Filters

When requesting datasets for ZIP processing via the catalog, it may be desirable to filter out categories of files. In addition to the dataset name masking characters (?, *, and **), ***PKZIP for MVS™*** provides the following commands to limit cataloged file selections:

`-EXCLUDE(dsname|mask)` - This command is used to avoid selecting datasets based on the file name. Multiple EXCLUDE commands may be specified for an individual ZIP call.

-SELECT_DSN_ALIAS(N) - This command is used to avoid selecting datasets based on a catalog ALIAS definition.

-SELECT_TAPE(N) - This command is used to avoid processing tape files.

-SELECT_VSAM(N) - This command is used to avoid processing VSAM Clusters (this does not affect the Archive Dataset organization). The Archive may be VSAM, while the clusters are excluded for ZIP processing.

-SELECT_MIGRATED(N) - This command is used to avoid processing DISK files that have been migrated using a product such as IBM's DFSMShsm. Files in this category are identified in the catalog as having a volume serial of "MIGRAT".

-SELECT_GDGALL - This command is used to allow ***PKZIP for MVS™*** to select all generations of a Generation Data Group, while **-SELECT_NOGDGALL** disables this feature (these are synonyms for the -GDGALL_SUPPORT(Y|N) command).

-RECURSE_LEVELS(N) - This command specifies if lower level dataset name masking is not desired.

INFILE DD Requests

When requesting a dataset for inclusion in ZIP processing via –INFILE (with an associated JCL DD statement), operating system allocation is performed before ***PKZIP for MVS™*** execution begins.

JES2 SYSIN INFILE Support

JES2 SPOOLed input data is supported for input ZIP processing. By referencing a "//… DD * "statement with an INFILE command, the input stream will be treated as a sequential file with DCB attributes of RECFM=FB, LRECL=80, and BLKSIZE=80. The filename generated will be based on the DSN generated by the JES2 subsystem and will be modified to end in "SYSIN", for example, `userid/jobname/JOBxxxxx/sysinfo/SYSIN`. When performing a PKZIP operation against an existing Archive, the DCB attributes (LRECL, BLKSIZE) will now be retained in the new Archive unless explicitly overridden with new command values.

> **Note: When performing an EXTRACT of such a file, –OUTFILE_… space allocation and volume information must be provided through the defaults module or command input stream, since JES2 DD statements do not carry space attributes.**

Input ZIP Archive Files

During an –ACTION(FRESHEN) or –ACTION(UPDATE) request; files contained within the old ZIP Archive are added to the candidate list. Names as previously stored are used to search the system catalog for viability (any file names not found in the system catalog remain in the ZIP Archive).

During an –ACTION(COPY), only files within the input Archive are candidates for copying to the new Archive (which must be unique from the input Archive).

File Selection Processing Notes

Files are not normally opened during the file selection phase of processing in order to streamline performance. However, some file characteristics are gathered for non-tape files at this time. PDS and PDSE datasets are opened so that their directory information can be reviewed and members identified for selection.

&SYSUID may be used in cataloged dataset selection requests. Multiple components of ***PKZIP for MVS™*** are used to process File Selection requests. Various informational messages can be obtained from these internal components by turning on various logging and trace elements in the command stream. PDS member name selection can be requested through –INFILE command parameters, associated JCL DD member reference, or Dataset name parameters.

- When an INFILE JCL DD specification is used and a member-name is coded in the JCL, it overrides any –INFILE command parameters. (Only the member requested in the JCL will be added to the selection list).

- Dataset name command requests that also contain member request masks act in a cumulative fashion. All members from a PDS matching the requested masks will be added to the candidate member list.

- When both INFILE and Dataset Name command requests are made with member names, the multiple requests are merged into a cumulative list, and only one copy of the member is processed.

- Because member name selections can also be placed on Dataset name masked requests, such as, more than one dataset is identified via a masked name, combinations of requests may result in different member-selection criteria for different datasets.

- Member selection requests are compiled into an internal table, which is later used to match against the list of members available from the PDS. PDS members will be selected in alphanumeric order.

Cataloged Dataset Name and INFILE Request Restrictions

Cataloged dataset command requests must begin with a fully qualified first level. For example, `SYS1.**` is valid, but `SYS*.**` is not.

Cataloged Dataset name requests depend on the accuracy of the system catalog structure under which PKZIP for MVS is executing. If a dataset is cataloged, but does not exist on the cataloged device, then an allocation error will occur later in processing.

INFILE(ddname) requests must accurately reflect the device and volume for the requested dataset. “ddname” must be a fully-qualified DDname allocated to the job step (or TSO session).

INFILE requests, which refer to a DD statement that is a concatenated set of datasets should have all files of the same DSORG and RECFM in accordance with OS/390 rules for concatenated datasets. The associated DD statement will be opened with the DCB characteristics of the first file in the concatenation, and that file’s name will represent the group for processing in the ZIP archive.

Dataset `ALIAS` names may be used to identify candidate datasets. However, the system catalog structure will be used to translate the `ALIAS` name to the true dataset name for processing. When a Dataset name request is made, a message will be issued to the output log indicating that an `ALIAS` to Truename translation has occurred. However, when an `ALIAS` name is used with an INFILE request, the operating system resolves the `ALIAS` entry to its associated Truename before program execution begins, and File Selection will only refer to the Truename as presented by OS/390.

Generation datasets (GDG) can be requested with a fully-qualified generation name, for example, "SYS1.BACKUP.G0020V00", a relative generation level, for example, "SYS1.BACKUP(-1)", or a GDG-base request. In all cases, identified candidates will resolve to their fully qualified NONVSAM dataset name, with each being processed as an independent entry.

- GDG-base selection only applies to ZIP processing at the time of the request in accordance to the current catalog structure.
- Relative generation selection is valid only with –INFILE and JCL specifications.
- UNZIP processing requires selection according to fully qualified generation names.

When GDG-base names are used via Dataset name command requests, each current ASSOCIATION entry in the catalog will be used to identify individual NONVSAM entries, and each will be processed as an independent entry. This differs from the way GDG-base names are handled when INFILE is used.

When an INFILE request is used in conjunction with a DD statement to reference a GDG-base, standard MVS expansion of the GDGALL name will occur. This results in all generations being treated as a concatenation group, with the latest generation name being assigned to the file. You must take care in handling the resultant ZIP file, since the data from 1 or more generations are included in the file. This differs from the way GDG-base names are handled when Dataset name requests are made.

VSAM files are supported at the CLUSTER level only. Individual DATA and INDEX COMPONENT names should not be requested.

ZIP File Names

The ZIP Archive architecture describes files in an internal format that is comparable to the UNIX file naming standards. Each file is described within a ZIP Archive Central Directory entry, and is represented in ASCII. In addition, the format carries an inherent directory/sub-directory format (with "/" as the directory separator character).

MVS dataset names are converted to the standard ZIP Archive file directory format before they are stored. For example, the dataset "SYS1.PARMLIB(CLOCK00)" will appear in a ZIP Archive as "SYS1/PARMLIB/CLOCK00". In addition, a browse of the file in HEX format would show the ASCII representation for the characters, not EBCDIC.

The following commands are used to control the file names being saved and restored during ZIP and UNZIP processing: (See the appropriate command section later in this manual for more detail).

Summary of Commands Affecting ZIP Filename

Process	Command	Description
ZIP & UNZIP	–TRANSLATE_TABLE_FILEINFO	EBCDIC <=> ASCII translate table
ZIP & UNZIP	–ZIPPED_DSN_SEPARATOR	Default is "/" and replaces "." In MVS DSNs, as well as separating a member name.
UNZIP	–UNZIPPED_DSN	Allows the transformation of the internal ZIP Filename to an MVS standard name and allows the replacement of qualifiers during the process.

Process	Command	Description
ZIP	–ZIPPED_DSN	Allows the transformation of the MVS DSN to an internal ZIP Filename.
ZIP	–PATH	Specifies whether the higher-level qualifiers should be stored as a directory pathname in the ZIP Filename.
UNZIP	–HIERARCHY	Determines what should be done with the hi-level qualifiers (directory path structure) of the ZIP Filename during the conversion process.
UNZIP	–FILE_EXTENSION	Specifies what should be done with a low-level extension (such as .TXT) during an EXTRACT request.
ZIP & UNZIP	–SIMULATE(Y)	Provides a means of running a simulation to determine what the resulting names will be.

Essentials for running PKZIP and PKUNZIP

PKZIP can perform various actions according to one of the following commands:

```
[ -ADD | -COPY | -DELETE | -FRESHEN | -UPDATE | -VIEW ]
```

The actions are described below. –ADD is the default action if none of the above actions are specified.

Command	Description
–ADD	Used to add files that are not already present into a new or existing ZIP archive.
–COPY	Used to copy a subset of an archive to a new archive.
–DELETE	Used to delete selected files from an existing ZIP archive.
–FRESHEN	Used to update existing files in an existing ZIP archive.
–UPDATE	Used to add new files to or update existing files in an existing ZIP archive.
–VIEW	Used to display details of selected files in an existing ZIP archive.

Each of the actions requires a ZIP archive to process so the –Archive command (or –Archive_OUTDD) must always be specified.

```
-ARCHIVE(<ZIP dataset name>)

-ARCHIVE_DSNAME (<ZIP dataset name>)
```

Finally you must specify the dataset(s) to be added, copy, deleted, freshened, updated, or viewed in the archive. You can do this using standard MVS dataset naming, for example:

```
MY.INPUT.DATA.SEQ
```

This line identifies a single file that is to be processed by PKZIP.

PKUNZIP

PKUNZIP is used to extract a number of compressed datasets (ZIP files) from a zip archive. To achieve this, PKUNZIP needs to be told three things:

1. The action to perform.
2. The archive from which the datasets are to be decompressed.
3. The files that are to be extracted from the archive.

PKUNZIP can perform various actions according to one of the following commands:

```
[ -EXTRACT | -TEST | -VIEW ]
```

The actions are described below. –EXTRACT is the default action if none of the above actions are specified.

Command	Description
–EXTRACT	Used to extract selected files from an existing ZIP archive.
–TEST	Used to delete selected files from an existing ZIP archive.
–VIEW	Used to display details of selected files in an existing ZIP archive.

Each of the actions requires a ZIP archive to process so the command (or alternative) must always be specified.

```
-ARCHIVE(<ZIP dataset name>)

-ARCHIVE_DSNAME (<ZIP dataset name>)
```

Finally, if a subset of all files in the archive are to be processed, you must specify the dataset(s) to be extracted, tested or viewed in the archive. You can do this using standard MVS dataset naming (See Note) or internal zip file naming conventions, for example:

```
MY.INPUT.DATA.SEQ

MY/INPUT/DATA/SEQ
```

The default is to select all files from the archive.

Note: To process an MVS DSN format for PKUNZIP selection, the name must readily match the internal zip name with the exception of the directory separators, such as, substitutes for "/", and the target MVS name must be acceptable to the operating system. (See –OUTFILE_DD and –UNZIPPED_DSN).

Chapter 6. - ZIP Files

Data Formats - Text or Binary

Data files are held within a ZIP Archive in either text or binary format. Both formats are supported by ***PKZIP®*** compatible products on other platforms, however, some restrictions apply to cross platform usage of the data, for example, workstation-based applications may not be able to process EBCDIC-based data that is commonly produced by S390 platforms.

Text data is represented by one of two character sets: EBCDIC or ASCII, in which individual alphanumeric characters are assigned an internal numeric code within the range of 0-255 (hexadecimal 00-FF). Although most of the same characters, for example, A-Z, a-z, 0-9 are contained in the EBCDIC and ASCII character sets, different code assignments are used for each. To preserve cross-platform compatibility of files containing only text characters, the –DATA_TYPE(TEXT) or –DATA_TYPE(DETECT) commands should be used. This will direct ***PKZIP for MVS™*** to translate EBCDIC characters into the ASCII character set (which is the standard set used by ***PKZIP®*** compatible products).

The –DATA_TYPE(BINARY) command is used to direct ***PKZIP for MVS™*** to bypass EBCDIC to ASCII character translation. This feature is useful when the file contains non-text data (**Warning:** Binary fields may generate what appear to be record delimited characters). Therefore, –TEXT should not be used if binary data is present. (Note that a custom TRANSLATETABLE_DATA table can be built to change control characters (X'0D' + '25' EBCDIC) to blanks) (such as graphics or internal numeric representations, for example, packed, or binary numeric data), or if text-based data is to be extracted only to other EBCDIC based platforms.

All data within a file is treated the same during ZIP processing in accordance with the –DATA_TYPE(TEXT) and –DATA_TYPE(BINARY) commands. Care should be taken when zipping files that may contain both text and binary data. Use of the –DATA_TYPE(TEXT) command when binary data exists within the file will produce unpredictable results for fields containing binary data.

–DATA_TYPE(BINARY) should be used to preserve data integrity; however, this means that text data will not be translated into the ASCII format by UNZIP processing in a cross platform environment.

An advanced feature; –DATA_TYPE(DETECT) is provided for ZIP processing, which instructs, ***PKZIP for MVS™*** to read a portion of data from the input file (in accordance with the –DATATYPE_DETECT_DEPTH value) and scan it for non-translatable text characters using the active text translation table. If the number of translatable text characters (as specified by the –DATATYPE_DETECT_TABLE) meets or exceeds the percentage specified by –DATATYPE_TEXT_PERCENT, the file will be treated as –DATA_TYPE(TEXT). Otherwise, it will be treated as if DATA_TYPE(BINARY) has been used. Note: One exception to this is X'00', or the NULL terminator character, which is commonly used in C language. The NULL character will be allowed within the files. If it is unknown whether a file in the ZIP Archive is text or binary, you may use the –ACTION(VIEWDETAIL) command to examine the file attributes.

It is possible for members of the same PDS or PDSE to be treated differently when –DATA_TYPE(DETECT) is used because of a varying mix of data. Each member is treated as an independent file during ZIP processing.

An advanced feature; –DATA_TYPE(DETECTX) is provided for UNZIP processing to assist in identifying and translating text-based files. This is useful when the originating ZIP platform (typically a workstation) does not set the "text" indicator for the file in the Archive.

Data Format - Text Records

In the context of ZIP Archives, a “text file” is one that is stored in the ASCII format. A text file contains records of data, each separated by a delimiter to signify the end of the record.

> **Note: An EBCDIC file containing text information (such as source code) can be stored in its original format by using –DATA_TYPE(BINARY), but it is not considered to be a “text” file within the ZIP architecture.**

PKZIP for MVS™ uses the default delimiter CR-LF (x’0D0A’) at the end of each text record. You may choose to use a different delimiter by using the –DATA_DELIMITER command (or other characters as specified in the command set). At the end of each ZIP’d file is a file terminator. The default file terminator for ***PKZIP for MVS™*** is Ctrl -Z (x’1A’). This file terminator can be changed by using the –FILE_TERMINATOR command.

> **Note: The last record will have the data delimiter followed by the file terminator.**

If it is desired that the ZIPPED file contain no data delimiters, you may specify –CRLF(N) or –DATA_DELIMITER(). If CR-LF is specified on ZIP, but –CRLF(N) is specified on UNZIP, then ***PKZIP for MVS™*** will treat the x'0D0A' as data characters, and they will be translated into the EBCDIC equivalent and be imbedded in the output file. Although it is possible for fixed length records to be properly aligned in an output file without CR-LF (by using input and output files with identical record lengths), care must be taken when using –CRLF(N) because –DATA_DELIMITER is the only explicit mechanism available to determine record lengths for text files.

At the time of UNZIP file extraction, ***PKZIP for MVS™*** will change text data from ASCII to EBCDIC by using a translation table. During installation, several translation tables are available, and the customizing process selects one of the translation tables as default. Additional translation tables may be created through the customizing procedure.

It should also be noted that during UNZIP processing, if the defined CR-LF character sequence, for example, x’0D0A’ is not found in the scan of the first buffer of data, the PKUNZIP program will attempt to locate a valid record terminator character to use throughout the extraction of that file.

> **Note: Unpredictable results may occur if a mix of the control characters X’ OA’, X’ OD’, or X’ 1A‘ are found in the input stream. PKZIP will use the first occurrence of these characters when automatic detection is used.**

For example, if a ZIP Archive had been brought from a standard UNIX platform, the record delimiter would have been saved as x’0A’. UNZIP processsing will dynamically re-define the –DATA_DELIMITER value for the remainder of that file. This is also useful if multiple ZIP Files are contained within the same Archive and have differing record delimiters.

Situations may arise in unique platform interchanges or when working with text files from different countries when the default translation table is not adequate. You may select any available translation table by using the –TRANSLATE_TABLE_DATA command.

Note: The *PKZIP for MVS*™ INSTLIB contains sample JCL and source members to assist in creating customized translate tables.

***PKZIP for MVS*™** extracts text records stored in the ZIP Archive by examining the data for record delimiter and file terminator indicators. Using these indicators, ***PKZIP for MVS*™** aligns records in accordance with the target file attributes.

Data Format - Binary Records

Binary data is stored in the ZIP Archive in its original format. Binary data may be graphics or numbers that are already in "computer format;" therefore, no translation is done. The length of binary records in UNZIP processing is determined in one of two ways:

- Fixed-length records, ***PKZIP for MVS*™** will automatically fill the available block according to the
  allocation specifications.

- Binary records of variable length, a Record Descriptor Word (RDW) will be
  inserted via the–SAVE_LRECL(Y) command. An indicator is tracked in the Archive directory that instructs UNZIP processing to automatically use these lengths when extracting the file. Use of this
  feature is extremely important when processing binary data with varying-length records. Note that the record length is in little-Endian format within the archive, not S390 format.

File Attributes

Within the ZIP Archive are two different directories providing information about the files held within the archive. A Local Directory is included at the front of each file, with information pertaining to it, for example, file size and date ZIPPED. The Central Directory is located at the end of the ZIP Archive. The Central directory lists the complete contents of the ZIP Archive and is the primary source of information for controlling UNZIP processing.

***PKZIP for MVS*™** will optionally store extended attributes about the file that can be useful in re-creating the file during UNZIP processing. These attributes include items such as space allocation, maximum record size, dataset organization (VSAM/PDS/SEQ, etc.). Additionally, an optional sub-category of extended attributes is available. Extended attributes for NONVSAM files include record format, DSORG, LRECL, and block size. Extended attributes for VSAM files would include CLUSTER information. File attributes can be displayed by using the –ACTION(VIEWDETAIL) command.

***PKZIP for MVS*™** enables you to store the extended attributes in the local directory, central directory recommended, both, or neither. See the command reference section for the specific command for each of these options. Attributes held in the Central Directory are used by PKUNZIP.

Large File Considerations

It is best when using the ZIP process for large files to use half-track blocking for the ZIP Archive (this is the default size employed by ***PKZIP for MVS*™**). This method provides the best performance and makes the most efficient use of storage space for both ZIP Archives and ZIP temporary files. Usage of other block sizes will decrease the volume of data that can fit onto a single volume, in addition to affecting performance levels.

A temporary work file may be created during the updating or reconfiguring of a file in the ZIP Archive, depending on file size and available storage. This temporary file may or may not have the same storage attributes as the original file. The temporary file holds the updated form of the file in order to allow for the reformatting of the (new) ZIP Archive. To preserve the integrity of the original Archive in case of a failure, the old Archive is preserved while a new Archive is being built. Therefore, there must be enough space allowed to accommodate the size of the old Archive, the temporary file, and the updated Archive.

Determining File Size

Default space allocations may not be adequate in the compression of large files. To calculate the space needed for the ZIP Archive and the temporary files, the following proportions may be helpful:

ZIP Archives - Primary: 25% (one-quarter) of the total size of the uncompressed file(s) (–ARCHIVE_SPACE_PRIMARY command).

ZIP Archives - Secondary: 10% (one-tenth) of the total size of the uncompressed file(s) (–ARCHIVE_SPACE_SECONDARY command).

Temporary Files - Primary: 25% (one-quarter) of the size of the largest uncompressed file (–TEMP_SPACE_PRIMARY command).

Temporary Files - Secondary: 10% (one-tenth) of the size of the largest uncompressed file (–TEMP_SPACE_SECONDARY command).

If a tape-based Archive is being used, it is possible to use a temporary disk archive during processing (see –STAGE_TAPE_ON_DISK command). The sizes used should correspond to those specified in the tape archive.

Chapter 7. - File Processing

File Support

PKZIP for MVS™ has the ability to support files maintained in various formats; specifically, sequential files, PDS, or PDSE members, VSAM files, and magnetic tapes or cartridges. ***PKZIP for MVS™*** has three possible applications for each file type:

Compressing files of each format into a ZIP Archive.

Data from a ZIP Archive may be extracted into each of these formats.

A ZIP Archive may be managed in each of these formats.

An overview of information regarding each file type is shown in the table below. Additional information that is required in working with each specific file type is detailed under the appropriate section later in this chapter.

In all cases, ***PKZIP for MVS™*** will optionally save file type information during ZIP processing. This information may be used by ***PKZIP®*** compatible products in applicable environments for an equivalent reconstruction of the file during UNZIP processing.

Licensing

Note that separate “file handler” licenses are required to process various types of datasets, for example, PDSEs, Tape, etc. These licenses also apply to the types of datasets being used as archives. For example, the VSAM file handler license is required to read VSAM ZIP files and to create a VSAM archive. See Appendix A - Licensing Requirements for more details.

	Sequential Files	PDS or PDSE Members	VSAM Files	Magnetic Tapes/Cartridges
Supported Record Formats	• Undefined: U • Fixed: F, FA, FB, FM, FBA, FBM, FBS • Variable: V, VA, VB, VM, VBA, VBM, VS, VBS[1]	• Undefined: U • Fixed: F, FA, FM, FBA, FBM • Variable: V, VA, VB, VM, VBA, VBM, VS, VBS	• ESDS • KSDS • RRDS	• Same as sequential files for standard-label and non-label tapes.
Supported ZIP Archive Formats	• Undefined: U • Fixed: F, FB, FBS • Variable: V, VB	• Undefined: U • Fixed: F, FB • Variable: V, VB	• ESDS	• See Magnetic Tapes/Cartridge section later in chapter.
File Selection Methods	• File name • File masks • JCL DD cards • ALIAS Path Name	• File name • File masks • JCL DD cards	• Cluster name • Path name • File masks • JCL DD cards	• JCL DD cards (see DD commands used with sequential files). • File names (limited to ZIP processing of cataloged tape files where mount authority is provided).

A. Sequential Files

In this chapter, the term "sequential file" is defined as being an MVS NON-VSAM dataset with DSORG=PS. This includes individual members of a GDG.

Note: Spanned record support has been included to assist in handling IEBCOPY unloaded files. This implies that you will ZIP the file with record length (SAVE_LRECL=Y) and binary (DATA_TYPE=BINARY) format.

Compressing Sequential Files

Batch jobs may be submitted to process sequential files using JCL DD cards, and/or by file selection specifications via control statements. The –INFILE command is used to reference a dataset that is allocated to the job step via a JCL DD statement. This directs ***PKZIP for MVS™*** to place the specified file into the Archive. Multiple –INFILE control statements may be used in a single execution. The files will be selected for processing in the order specified by –INFILE (not by the sequence of the JCL statements).

```
//MYFILE  DD  DISP=SHR,DSN=SYS1.PARMLIB(CLOCK00)
//SYSIN   DD  *
-ADD
-INFILE(MYFILE)
/*
```

Extracting Records into a Sequential File

The –OUTFILE command is used to reference a dataset for extraction into a sequential file format.

```
//TARGET  DD  DISP=(NEW,CATLG),DSN=userid.MY.SEQUENTIAL,UNIT=SYSDA,
//  SPACE=(CYL,(1,1)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920)
//SYSIN   DD  *
-EXTRACT
-OUTFILE(TARGET)
```

```
-ARCHIVE(MY.ARCHIVE)
/*
```

Managing a Sequential File ZIP Archive

A new sequential archive may be created by use of the –ARCHIVE_OUTFILE command with appropriate DCB information in the referenced JCL, or implicitly by specifying –ARCHIVE_DSN(ZIP_name) with –ARCHIVE_DSORG(PS).

```
//newarch  DD  DISP=(NEW,CATLG),DSN=userid.MY.ZIP,UNIT=SYSDA,
//  SPACE=(CYL,(1,1)),DCB=(RECFM=FB,LRECL=27998,BLKSIZE=27998)
//SYSIN  DD  *
-ADD
-ARCHIVE_OUTFILE(newarch)
userid.MY.JCL(*)          <= file to be ZIP'd
hlq.*.ASM(*)
```

Additionally, an existing archive may be read by use of the –ARCHIVE_INFILE command.

Processing GDGs

GDG members are generally treated as individual sequential datasets with their respective fully qualified names. With some restrictions, full GDGs and relative generations may be selected for ZIP processing.

The compression and extraction of GDGs (Generation Data Groups) present unique concerns. These are described in more detail in Chapter 5. - File Selection and Name Processing, in the section “Cataloged Dataset Name and INFILE Request Restrictions”.

File Concatenation for ZIP Processing

It is possible to concatenate multiple files of like attributes, such as, the same RECFM and LRECL, via –INFILE. These may include sequential files (DSORG=PS), fully qualified or relative generations of a GDG, or PDS/PDSE members.

It is important to note that ***PKZIP for MVS™*** will process the entire concatenation as one file stream, and use the first DSNAME in the concatenation sequence as its basis for saving file attributes in the ZIP Archive.

B. PDS and PDSE Members

Partitioned Datasets have a variety of unique characteristics and applications. For this reason, separate sections are dedicated to the following topics:

Selecting PDS/PDSE members for compression.

Extracting data into a PDS.

Managing ZIP Archives as PDS members.

Load libraries.

Selecting PDS Members for Compression

PKZIP for MVS™ operates on individual PDS members as distinct file entities, although a complete PDS or subset of a PDS can be operated on through JCL and control card specifications.

Note: Within this section, unless specified otherwise, the term PDS also applies to PDSE.

File Name or File Mask

PKZIP for MVS™ is able to compress a single PDS member, multiple PDS members, or all members of one or multiple PDS files by adapting the file selection name. Examples of these options are shown below.

```
//member1 DD DISP=SHR,DSN=SYS1.PARMLIB(CLOCK00)
//SYSIN  DD *
-INFILE(member1)

SYS1.PARMLIB(CLOCK00)  <= get a single member by catalog
SYS1.PARMLIB(CLOCK*)  <= get all members starting with "CLOCK"
SYS1.PARMLIB or SYS1.PARMLIB(*)  <= get all members
SYS1.PARMLIB(*00)  <= all members suffixed with "00"
MY.PDS(A??SRC)  <= any character in 2nd and 3rd positions
```

DD Statements

Batch jobs may be submitted to process PDS members using JCL DD cards. To process only one PDS member, the member name may be used as the file identifier. To process all members of a PDS, the PDS name may be used as the file identifier. To process several members, the –INFILE command is used along with the selected member names, or a file mask may be used in place of specific member names.

```
//pds DD DISP=SHR,DSN=SYS1.PARMLIB
//SYSIN  DD *
-INFILE(pds,CLOCK*,*00,MEMBER6)   <= multiple INFILE statements may be used.
```

Extracting Data into a PDS

PKZIP for MVS™ allows you to extract files from an Archive into either a new or existing PDS. A PDS member that has been compressed into the Archive may be extracted into a different PDS. In this case, file attributes for the target PDS can be governed by pre-allocation, JCL, control cards, or extended attributes previously saved in the Archive during ZIP processing.

At the time of extraction, the –OUTFILE_DSNTYPE(PDS) command is used to instruct ***PKZIP for MVS™*** to dynamically allocate a PDS from the specified files. The PDS name is governed by the use of –UNZIPPED_DSN, –FILE_EXTENSION, and –HIERARCHY(N).

```
//SYSIN  DD *
-ARCHIVE(my.zipfile)
-EXTRACT
-OUTFILE_DSNTYPE(PDS)
-OUTFILE_RECFM(FB)
-OUTFILE_LRECL(80)
-OUTFILE_BLKSIZE(27920)
-OUTFILE_SPACE_TYPE(CYLINDERS)
-OUTFILE_SPACE_PRIMARY(2)
-OUTFILE_SPACE_SECONDARY(1)
MY/PDS/MEMBER1  <= this is the archive filename selection to result in
MY.PDS(MEMBER1)
```

Managing ZIP Archives as PDS Members

PKZIP for MVS™ can maintain a ZIP Archive as a PDS member using the –ARCHIVE_DSN command along with the PDS and member name. When the Archive is created as a member of an existing PDS, the attributes for the PDS will not be altered.

Load Libraries

In most cases, load libraries will only be extracted to another OS/390 platform; therefore, ***PKZIP for MVS™*** is able to process either an individual member or an entire load library. The methods used will vary, as described below.

Processing Individual Members

Each member of the PDS is maintained as an individual file in the ZIP Archive. Both –DATA_TYPE(BINARY) and –RDW commands should be used to insure data integrity. In addition to normal data storage, necessary load module directory information is saved in the extended attributes section of the Archive directory. During extraction, any individual member may be selected for processing. When recreating the member on extraction, additional information (such as the TTR entry point) is translated by ***PKZIP for MVS™*** to use when loading the file.

Load Module Control

Some information, for example, the NOTELIST used for Overlay segments, is not retained in the Archive. This may cause inaccuracies upon extraction, as that load module may not be properly restored. To avoid this problem, it is recommended that the load module be placed in a library by itself, and that the file be extracted to a library with the same blocksize, on the same device type, or use the process described below.

Processing Entire Load Library

If it is not necessary to select individual members for later extraction, or if the library contains overlay segments or other specialized load modules, an alternate method is recommended. First, unload the PDS to a sequential file format supported by ***PKZIP for MVS™*** (such as IEBCOPY, or the TSO command TRANSMIT, which can be run in batch). Once this step is complete, ZIP the sequential file using normal ***PKZIP for MVS™*** processing. On extraction, ***PKZIP for MVS™*** will recreate the sequential file, which can then be reloaded to the PDS with the utility used previously. Although this method entails an extra step, it allows compression of the entire library, and there are no restrictions placed on individual members of the library.

See pkzip.mvs.INSTLIB(IVPVSPAN) for a sample job stream.

C. VSAM Files

VSAM files are selected and allocated with the use of the IBM Access Method Services utility IDCAMS, as described in the IBM Access Method Services manual. A working knowledge of IDCAMS processing will enhance the effectiveness of managing VSAM datasets with ***PKZIP for MVS™***. Control statements and input file characteristics are used by ***PKZIP for MVS™*** to internally generate Access Method Services control statements for dynamic calls to IDCAMS.

PKZIP for MVS™ makes use of Access Method Services User I/O Routines for SYSIN and SYSPRINT file requests. OEM products and/or Installation-written routines that modify standard IBM processing for these exits should not be active for PKZIP processing.

A sample JOB to demonstrate a ZIP and UNZIP of a VSAM KSDS to a VSAM Archive can be found in pkzip.mvs.INSTLIB(IVPVSAM).

Compressing a VSAM File

The cluster name is used when selecting a VSAM file for compression. Attempting to use only the data or index components of the file would most likely result in an unusable file. As with sequential and PDS files, either –INFILE (with JCL) or file selection statements may be used to identify VSAM files for processing.

VSAM files often contain a mixture of text and binary data. Therefore, unless it is necessary to translate the data to ASCII, both the –DATA_TYPE(BINARY) and –SAVE_LRECL commands should be used.

During ZIP processing, ***PKZIP for MVS™*** will determine the type of VSAM file requested from the system catalog. Through the use of –ATTRIB commands, this information can be retained in the ZIP Archive for use during UNZIP processing to reconstruct the cluster.

VIEWDETAIL of a KSDS in an Archive

The following –VIEWDETAIL shows the ZIP result of a KSDS file:

```
-ACTION(VIEWDETAIL)
-ARCHIVE(PKZIP.MVS.IVP.TEMP)
ZPAM012I ZIP comment: PKZIP for MVS™ by PKWARE of Ohio, Inc.
ZPAM013I ************************************************************
ZPAM001I Filename: PKZIP/TEST/KSDS
ZPAM002I File type:          BINARY  SAVED_LRECL (RDW)
ZPAM003I Date/Time:          10-MAY-2001 11:27:48
ZPAM004I Compression Method: STORED
ZPAM005I Compressed Size:    252
ZPAM006I Uncompressed Size:  252
ZPAM007I 32-bit CRC:         12CC21C8
ZPAM008I Created by:         PKZIP for MVS™   5.5 (PKZIP 2.x compatible)
ZPAM009I Needed to extract:  PKUNZIP     1.0
ZPAM301I File Type:          VSAM
ZPAM307I File Record Size: 100
ZPAM309I File Volume(s) Used: TSO001
ZPAM331I VSAM Cluster Type: INDEXED
ZPAM331I VSAM Cluster Catalog Name: CATALOG.TSO.USERCAT
ZPAM331I VSAM Cluster Erase: ERASE
ZPAM331I VSAM Cluster Format: INDEXED
ZPAM331I VSAM Cluster Free CI Space %: 33
ZPAM331I VSAM Cluster Free CA Space %: 10
ZPAM331I VSAM Cluster Imbed: NOIMBED
ZPAM331I VSAM Cluster Key Length: 10
ZPAM331I VSAM Cluster Key Position:  4
ZPAM331I VSAM Cluster Ordered: UNORDERED
ZPAM331I VSAM Cluster Avg. Record Size: 80
ZPAM331I VSAM Cluster Max. Record Size: 100
ZPAM331I VSAM Cluster Recovery/Speed: RECOVERY
ZPAM331I VSAM Cluster Replicate: NREPL
ZPAM331I VSAM Cluster Spanned: NONSPANNED
ZPAM332I VSAM Data    Name: PKZIP.TEST.KSDS.DATA
ZPAM332I VSAM Data    Type Space: CYL
ZPAM332I VSAM Data    Primary Space: 5
ZPAM332I VSAM Data    Secondary Space: 2
ZPAM332I VSAM Data    Buffer Space: 37376
ZPAM332I VSAM Data    CI Size: 18432
ZPAM332I VSAM Data    Reuse: REUSE
ZPAM332I VSAM Data    Share Options: 1,3
ZPAM332I VSAM Data    Volume: TSO001
ZPAM333I VSAM Index   Name: PKZIP.TEST.KSDS.INDEX
ZPAM333I VSAM Index   Type Space: TRK
ZPAM333I VSAM Index   Primary Space: 6
ZPAM333I VSAM Index   Secondary Space: 3
```

```
ZPAM333I VSAM Index   CI Size: 512
ZPAM333I VSAM Index   Reuse: REUSE
ZPAM333I VSAM Index   Share Options: 1,3
ZPAM333I VSAM Index   Volume: TSO001
```

Extracting Data into a VSAM File

It is helpful before extracting data from a ZIP Archive to be aware of what file name and file attributes are being stored for the compressed file. –VIEWDETAIL may be used on the Archive to verify the information. Unless –SAVE_FILE_ATTRIBUTES(NONE) is specified, the PKZIP program will save the cluster definition information in the Archive Directory. When the PKUNZIP program is run to dynamically recreate the file during EXTRACT processing, it will use the stored file characteristics to define the cluster unless overridden in the control cards. (This includes volume information, so Archives being transferred from one system to another, or being restored from an older environment may require –VSAM_DATA_VOLUMES override commands to avoid allocation problems to non-existent volumes.) Care must be taken when defining or overriding VSAM cluster specifications. Items such as MAX LRECL (the second parameter of –VSAM_RECORDSIZE) must be correct in order for the PKZIP program to correctly UNZIP the data to the target cluster.

When extracting records for insertion into a VSAM cluster, the PKZIP program will open the cluster in Load-Mode and attempt a sequential insert strategy. However, if a record key is rejected by VSAM PUT because it is out of sequence, the PKZIP program will change to direct-insert strategy for all subsequent records. This has the potential for two negative impacts:

- Performance may be severely impacted for large files.
- Because VSAM handles CI and CA splits differently for direct inserts, the cluster may expand beyond anticipated space requirements, thereby requiring a subsequent re-org or the extraction may fail due to space constraints.

For these reasons, if a large file is being extracted to a keyed VSAM cluster and the source data is not known to be in key sequence, then the following procedure is recommended:

Extract the file to a sequential dataset.

Sort the sequential file by the key field.

Use IDCAMS REPRO to load the target cluster.

Standard VSAM PUTs are performed during UNZIP operations. VSAM operating characteristics and limitations will be encountered (such as found during IDCAMS REPRO processing). A common occurrence may be that the defined VSAM CLUSTER may not have sufficient space to load the data due to FREESPACE designations. ***PKZIP for MVS™*** will report VSAM error and reason code information when these types of events occur.

To Overwrite a Current VSAM File

When extracting a compressed file to an existing VSAM file, it may be desirable to overwrite the existing file. By using the combined commands of –OVERWRITE and –VSAM_REUSE, the compressed file will replace the contents of the current file. File attributes will not be changed when processing a file overwrite, so you must assure the compatibility of the compressed file with the file being overwritten.

Note: In accordance with IBM's rules for REUSABLE clusters, the target cluster must have been defined with the REUSE attribute, otherwise, the open for the file will terminate with the message "ZPFM071E VSAM OPEN Error 000000E8 for File(ddname) DA(vsam_cluster_name)."

```
-ACTION(EXTRACT)
-OVERWRITE
-VSAM_REUSE(Y)
filename_to_be_restored
```

To Restore a Compressed VSAM File

PKZIP for MVS™ will retain the attributes of a VSAM cluster in the ZIP Archive unless otherwise specified. Upon extraction, the file attributes are used to recreate the VSAM file if there is not already an existing file. File attributes can be overridden during extraction by use of commands beginning with –VSAM_, –VSAM_DATA_, and –VSAM_INDEX_ as appropriate.

To Create a New VSAM File

A VSAM file can be created from a ZIP file even though the file was not originally a VSAM file, or the file attributes were unknown. By using the –MAKEVSAM command, along with any suitable –VSAM_… commands, a new VSAM file is created with the appropriate VSAM file attributes.

Using a combination of Archive File attributes, the ACZDFLT module defaults and any –SYSIN command overrides, ***PKZIP for MVS™*** will generate command input to IDCAMS similar to the example below.

```
DEF CL(NAME(PKZIP.MVS.IVP.KSDS) INDEXED    -
 BUFSP(37376)   CISZ(18432)                                    -
 ERASE    FSPC(33 10)  NONSPANNED   REUSE NOWRITECHECK              -
 RECSZ(80 100)  SHR(1,3)                                -
 VOL(TSO001                                                       -
) -
                 NOIMBED NREPL RECOVERY -
 KEYS(10 4) -
) -
 DATA(NAME(PKZIP.MVS.IVP. KSDS.DATA) -
 CYL(5 2)  ) -
 INDEX(NAME(PKZIP.MVS.IVP. KSDS.INDEX) -
 TRK(6 3)  -
 CISZ(512)                                  -
)
```

Note: *PKZIP for MVS™* may default selected commands from the ACZDFLT module, while IDCAMS may default some file attributes when they are not specified.

Managing a VSAM ZIP Archive

A VSAM Zip Archive supports the ESDS format. The –ARCHDSORG(VS) command is used to create the Archive. See pkzip.mvs.INSTLIB(IVPVSAM) for an example of creating a VSAM Archive.

Archive VSAM allocation specifications may be changed by using the –ARCHIVE_…and VSAM_…commands. The Access Method Services section of the IBM Manual on the DEFINE CLUSTER command may be consulted for more information.

To Update a VSAM ESDS ZIP Archive

In order to update a VSAM ZIP Archive, ***PKZIP for MVS™*** creates a new ZIP Archive, and then deletes the previous Archive. If either –ARCHTO or –ARCHFOR commands were used when the Archive was originally created, a problem may occur during the deletion process, as the retention period for the VSAM ZIP Archive may still be in operation.

To Process "Sparse" RRDS Files

PKZIP for MVS™ will use the same process as IDCAMS REPRO in processing VSAM RRDS files that contain unused "slots." In copying the RRDS to a sequential dataset, the missing slots are treated as nonexistent. If an RRDS is later created, any missing slots will not be included in the new file. As a result, the slot number of some of the copied records will be different than the original.

Therefore, ***PKZIP for MVS™*** will correctly recreate only those RRDS files with no interspersed empty slots. Variable length and fixed length RRDS files are both processed with this constraint.

Unsupported File Types

PKZIP for MVS does not directly support alternate index files or paths. A VSAM Alternate Index can be managed in two ways.

It is recommended that the base cluster be processed, and then the Alternate Index be recreated at the time of extraction.

Using the other option, the data may be copied to another supported dataset type using the Alternate Index, and the copy compressed. At the time of extraction, the process would be reversed. This maintains the data in the ZIP Archive in the same order as it was contained in the Alternate Index.

D. Magnetic Tapes and Cartridges

PKZIP for MVS™ can process cataloged tape files using file names (as specified in the table at the beginning of this chapter) or DD command. When an output file or a non-label tape file is defined by the DD command, it must include DCB information on the DD statement.

Copying a Tape-Based Archive to a Disk File

To enhance performance, ***PKZIP for MVS™*** can use a temporary dataset as an interim measure when reading a ZIP Archive from an existing cartridge or tape based archive (governed by the –STAGE_TAPE_ON_DISK(Y) command). This will be the normal method for reading a Tape (3420). –TEMP commands are used to specify the size and format of the temporary dataset. If default size options are chosen or if the ZIP Archive is very large, it is possible that the temporary dataset may not be large enough for the entire ZIP Archive. This situation would produce x37 abend errors, and the temporary dataset would be invalidated, causing ***PKZIP for MVS™*** to process the file directly.

Note: Specifically, refers to Magnetic Tape (3420 style) or Magnetic Cartridge (3480/3490 style). Unless differentiated in the context, the information in this chapter refers to both tape and cartridge.

The //ARCHTEMP DD is used for this procedure. Normally, ***PKZIP for MVS™*** will dynamically allocate this file, however, it is possible to allocate the DD statement directly in the JCL to provide manual control over

the allocation of the staging file. In addition, the ARCHTEMP file may be allocated as a permanent dataset. Using these techniques, the following additional benefits can be obtained:

The permanently staged archive may be used as a backup copy, for example, maintain GDGs of the Archive in a “before” picture.

Retain the disk-based archive for subsequent processing runs.

More information may be found in the command section on –STAGE_TAPE_ON_DISK.

Compressing Data from Tape

PKZIP MVS™ will process cataloged-Standard Label tape files in the same way that disk files are (either through dataset selection control cards or DD statements with INFILE). However, the file attributes that are stored with the Archive for the related file are limited to information such as LRECL, BLKSIZE,and RECFM. When extracting files such as this to disk, –OUTFILE_ commands should be provided either by command or the defaults module to specify proper space allocation information.

Non-labeled Tapes (NL)

Non-label tapes do not contain DCB information that is necessary for ***PKZIP for MVS™*** to process the compression (such as, RECFM, LRECL, and BLKSIZE). This is not an issue when using standard-labeled tapes, as the information is coded in the label. It is imperative that the required information be included in the DD statement, as shown in the example below, otherwise standard system OPEN abends will result.

```
//TAPEIN  DD DISP=OLD,DSN=my.tape.file,UNIT=TAPE,
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=32720)
//  LABEL=(1,NL)
//SYSIN DD *
-ARCHIVE(my.archive)
-INFILE(TAPEIN)
```

Restriction: Non-label (NL) tape datasets should not be selected via control cards, because the DCB information cannot be obtained for the file.

File Attributes

The minimal file attributes that are stored for tapes when compressed are DSORG, RECFM, LRECL, and BLKSIZE. These are apparent in the example of Archive detail as shown below:

–VIEWDETAIL Display

```
ZPAM012I ZIP comment: PKZIP for MVS™ by PKWARE of Ohio, Inc.
ZPAM013I ************************************************************
ZPAM001I Filename: userid/TEST/TAPE
ZPAM002I File type:          TEXT
ZPAM003I Date/Time:          10-MAY-2001 15:43:30
ZPAM004I Compression Method: DEFLATE -NORMAL
ZPAM005I Compressed Size:    34
ZPAM006I Uncompressed Size:  247
ZPAM007I 32-bit CRC:         9EBBDFBB
ZPAM008I Created by:         PKZIP for MVS™   5.5 (PKZIP 2.x compatible)
ZPAM009I Needed to extract:  PKUNZIP     2.0
ZPAM301I File Type:          NONVSAM  SEQUENTIAL
ZPAM303I File Record Format: FB
```

```
ZPAM307I File Record Size: 80
ZPAM308I File Block Size: 6160
ZPAM309I File Volume(s) Used: SC0016
ZPAM310I File Creation Date: 2001/05/10
```

Extracting Data onto Tape

PKZIP for MVS™ requires three components to extract data onto Tape.

Specify the ZIP file to extract using an appropriate file selection.

Use a DD statement to specify the Tape dataset you are extracting to, being sure to include DCB information.

Use the –OUTFILE command to extract the ZIP file onto the appropriate Tape, as specified in the DD statement.

Restriction: Only one –OUTDD statement can be used per job. It is recommended that datasets be extracted to tape one at a time.

Managing a ZIP Archive on Tape

PKZIP for MVS™ has the capability of reading or writing ZIP Archives on tape. The –ARCHIVE_INFILE and
–ARCHIVE_OUTFILE commands are used to specify the tape to be processed.

To Process Multiple-Volume Tape Archives

A Tape archive contains information at the end of the tape that is necessary for ***PKZIP for MVS™*** processing. ***PKZIP for MVS™*** will scan the tape until it finds the information, and then return to the beginning of the tape to begin processing. Because this necessitates accessing the tape at least twice, one of the following options should be considered to reduce the impact of the tape handling:

- Mount all the required tapes at once. This can be done by specifying the unit count parameter on the DD statement (keyword UNIT). For example, if two tape units are to be allocated, the DD statement would read UNIT=(TAPE, 2), thus insuring that both volumes of a 2-volume archive will be mounted.
- The UNIT= parameter for any tape file must match the devices defined for the local system. The systems programming staff at the installation should be contacted for information regarding these units and standards for use.
- Copy the tape archive to a disk file, and processing the disk instead of tape.
- Use –TAPETODISK command of ***PKZIP MVS™*** to copy the archive to disk.

To Compress Data into a ZIP Archive on Tape

By using the –ARCHIVE_OUTFILE command, ***PKZIP for MVS™*** will compress data into a ZIP Archive residing on tape. A DD statement is used to specify the new tape-based archive dataset, and should include necessary DCB information. The –ARCHIVE_OUTFILE command replaces any –ARCHIVE_… commands intended to dynamically create an Archive, and directs ***PKZIP for MVS™*** to create the ZIP Archive on the tape dataset as specified by the name in the DD statement.

```
//ARCHOUT DD DSN=hlq.archive.zip,UNIT=tape[2],DISP=(NEW,CATLG),
//  DCB=(RECFM,LRECL=32760,BLKSIZE=32760),LABEL=(1,SL)
//SYSIN  DD *
-ARCHIVE_OUTFILE(ARCHOUT)
```

To View a Tape-Based Archive

Tape-based Archives may be viewed in the same way as disk-based Archives. Either a DD statement referenced by –ARCHIVE_INFILE (with appropriate DCB information if the tape file is Non-label), or a cataloged Standard-label tape via –ARCHIVE.

Restriction: Some data centers do not allow dynamic allocation of Tape datasets. In this case, –ARCHIVE_INFILE should be used with a DD statement.

Processing Hint: If you intend to VIEW the archive and later process it for EXTRACTION, you may save the time of re-processing the tape volume(s) by specifying –STAGE_TAPE_TO_DISK with an //ARCHTEMP DD statement to direct the PKUNZIP program to create a disk copy of the archive for subsequent processing. The disk Archive can then be used for the EXTRACT (or further VIEWing with ISPF).

```
//ARCHTEMP DD DSN=permanent_dsn,DISP=(NEW,CATLG),UNIT=disk_device DD statement.
```

The sample JCL below demonstrates the creation of a ZIP Archive on tape, followed by a step to view the cataloged tape dataset.

```
//ZIPIT    EXEC PGM=PKZIP
//SYSPRINT DD SYSOUT=*
//ARCHOUT  DD DSN=&SYSUID..TAPE.ZIP,
//         DISP=(NEW,CATLG),
//         UNIT=(3490,,DEFER),
//         LABEL=(1,SL),
//         DCB=(RECFM=FB,LRECL=32760,BLKSIZE=32760)
//SYSIN    DD *
-ARCHIVE_OUTFILE(ARCHOUT)
-ACTION(ADD)
PKZIP.MVS.INSTLIB(DATASEQ1)
/*
//VIEWIT   EXEC PGM=PKUNZIP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
-ARCHIVE(&SYSUID.TAPE.ZIP)
-ACTION(VIEW)
/*
```

To Extract Data From a Tape-Based Archive

A tape-based archive can be specified via –ARCHIVE_INFILE, (along with necessary DCB information on the associated DD statement for a Non-label dataset), or with –ARCHIVE for a cataloged Standard-label dataset.

Performance note: Processing a tape-based Archive may be faster when specifying –STAGE_TAPE_TO_DISK(Y). The reasons for this are as follows:

The architecture of a ZIP Archive (on all platforms for all ***PKZIP***® 5.x products) has the Central file directory at the back of the Archive. This is also where some important file information is kept (such as whether the file is Text-needing translation, or Binary). Therefore, the PKUNZIP program must read the back of the Archive before scheduling the processing of the files, then rewind and read from the beginning.

Because of the serial-nature of the tape media, only 1 task can be used to EXTRACT the data. When many non-partitioned files are being selected for processing, multi-tasking may be beneficial when used in conjunction with a disk-based archive.

To Update Files in a Tape-Based Archive

PKZIP for MVS™ requires the use of a new tape to update files residing on a tape-based Archive. –ARCHIVE_INFILE and –ARCHIVE_OUTFILE must be used. The input and output Archives do not need to both be of the same media type (one may be disk and the other tape).

Chapter 8. - ZIP Archives

A ZIP Archive is the storage facility for files that are compressed, or simply stored using the ***PKZIP for MVS™*** product. It can hold up to 65,535 files which may have been compressed up to 99% of their original size. In addition, file attributes are retained allowing extraction of the same file characteristics without the need of control card specifications. Data integrity is validated by a Cyclic Redundancy Check (CRC) to maintain integrity from compression through extraction.

An Archive can exist in three possible states during processing. They are described as "old Archive," "temporary dataset" and "new Archive." An explanation of the functions of each of these is described in the sections below.

Many older ***PKZIP®*** products were modeled after the disk operating system (DOS)-based ***PKZIP®*** products, which had an archive limit of 16,383 files. The ZIP Archive specifications allow up to 65,535 files based on a 2-byte binary counter in the Directory. An Archive that is created by ***PKZIP for MVS™*** 5.5 with greater than 16,383 files may not be able to be processed by older releases of ***PKZIP for MVS™*** or ZIP products written by other vendors. The actual number of files that can be processed by ***PKZIP for MVS™*** is limited by local system resources such as allowable Region Size.

A ZIP Archive is transferable between platforms, for example, files that are compressed by ***PKZIP for MVS™*** may be extracted by ***PKZIP®*** on a different platform maintaining identical data.

MVS Archives can be held in a variety of formats: sequential dataset on tape or disk, PDS or PDSE members, or a VSAM Cluster (ESDS). An Archive file is designated to ***PKZIP for MVS™*** by a control card of either –ARCHIVE_DSN(dsname) or –ARCHIVE_INFILE(ddname).

Sequential dataset Archives may be held in Undefined (U), Variable (V, VB) or Fixed (F, FB, FBS) formats. PDS and PDSE member Archives may be held in Undefined (U), or Fixed (F, FB) formats.

The standard format for a ZIP Archive is shown in the table below:

Standard Zip Archive Format
File #1 [Local Directory Entry (X'504B0304')] [optional extended attributes][file data]
File #2 [Local Directory Entry (X'504B0304')] [optional extended attributes][file data]
File #n [Local Directory Entry (X'504B0304')] [optional extended attributes][file data]
File #1 [Central Directory Entry (X'504B0102')] [optional extended attributes]
File #2 [Central Directory Entry (X'504B0102')] [optional extended attributes]
File #n [Central Directory Entry (X'504B0102')] [optional extended attributes]
[End-Central Directory Entry (X'504B0506')] [optional Archive Comment]

The Local and Central Directory Entries contain information such as the file name, uncompressed size and compressed size, along with control values. The extended information controlled by the –SAVE_FILE_ATTRIBUTES command reflects dataset allocation information from the file as stored by ***PKZIP for MVS™***.

"Old" ZIP Archive

An old ZIP Archive refers to an Archive containing ZIPPED files that is in existence and may also be referred to as "ARCHIN". It may have been created by ***PKZIP for MVS™*** in an earlier process, or transferred from a different platform. This Archive is specified using the –ARCHIVE or

–ARCHIVE_INFILE commands. The old Archive can be thought of as the “before” version of an Archive that is being updated.

“Temporary” Dataset

A temporary dataset refers to a work in progress. This dataset has several possible uses in ***PKZIP for MVS™*** processing, including:

- The transient output Archive when an update action is requested against an existing Archive dataset.

  When a new non-partitioned Archive dataset is created by an update request, ***PKZIP for MVS™*** will use a temporary name for the output archive until the processing request is complete. Note that the system reports the cataloging of the temporary dataset name in the job log, not the final name used in the rename. This is normal behavior for dynamically allocated files in System/390 operating systems.

- As an interim storage area for compressed data before it is written to the output Archive.

  In addition to the Archive being allocated, temporary files may be allocated as staging areas for compressed data. The –TEMP family of commands governs the allocation controls for these temporary files.

- As temporary storage while processing tape input archives.

  The –STAGE_TAPE_TO_DISK command may be used to copy a tape Archive to a disk based temporary file to improve performance. ***PKZIP for MVS™*** will automatically process 3420 Reel to Reel tape in this way to accelerate the copying process. By manually defining //ARCHTEMP DD in the job, this temporary dataset can also be passed to subsequent PKUNZIP steps for better performance. The use of this method requires that the size of the temporary Archive be equal to or larger than the Archive.

- As temporary storage for file control information, including SORT work files.

  When a high volume of dataset names is encountered during catalog filename selection and Archive directory parsing, informational records may be written to work files for processing according to the memory controls provided in the job. Additionally, these temporary files are used for sort/merge processing for filename matching.

“New” ZIP Archive

When ZIP processing begins, **PKZIP MVS™** creates a new ZIP Archive that is the modified, or “after”, version of the old Archive. The (modified) name of the old Archive and specified allocation information of the old Archive is automatically transferred to the new Archive. After the update process completes, the old Archive is deleted. If the new output Archive allocation fails, ***PKZIP for MVS™*** will terminate, leaving the old input Archive intact.

Temporarily, the new ZIP Archive will keep the same name as the old ZIP Archive, as named in the –ARCHIVE command, except that the last part of the dataset name will be replaced by a unique 8-character name. If the new Archive is a member of a PDS or PDSE, this unique name acts as the member name.

Chapter 9. - Commands

Commands given to the ***PKZIP for MVS™*** programs are described in detail throughout this chapter with a summary of the available commands given below.

PKZIP for MVS™ can perform various actions in conjunction with the use of the following commands and modifiers:

[–ACTION(ADD|COPY|DELETE|EXTRACT|FRESHEN|UPDATE|VIEW)]

–ACTION(ADD) is the default action for ZIP processing and –ACTION(EXTRACT) is the default for UNZIP if none of the above actions are specified. The actions ADD, COPY, DELETE, FRESHEN, and UPDATE all make logical changes to an archive, while EXTRACT and VIEW only read an existing archive.

Each of the actions requires a ZIP archive to process so the following commands must always be specified:

–ARCHIVE_DSN(<ZIP dataset name>)

–ARCHIVE_INFILE(dd_name)

For details on how to input commands for processing by ***PKZIP for MVS™***, for example, SYSIN, PARM parameters, etc., refer to Command Details.

Command Syntax

Command strings and filenames are identified with delimiters of either a blank or semi-colon ";".

Non-blank characters found within a command buffer that are not identified as a command or comment, will be treated as a filename selection.

- Comments are currently supported when Column1 of an input buffer is an asterisk "*". Commands are identified by a hyphen "–" either in the first column of a non-continued line, or immediately following a blank or semi-colon. Unpredictable results will occur when unidentified characters are found in the input stream (depending on their location in the command structure).
- Command names will be accepted in mixed case.
- Command values which have specifically listed options will be translated to upper case for comparison, such as, mixed case allowed.
- Only selected commands values which are free-form in nature, for example, MVS file names, will be translated to upper case. Others, for example, internal ZIP filenames will retain case sensitivity.

Filename selections are case-sensitive.

File Selections vs. Commands

A ***PKZIP for MVS™*** command is defined by placing a "–" (hyphen) character in front of a valid command string. If no "–" character is found at the start of a sequence of characters, the characters are interpreted to be part of a file selection for ZIP or UNZIP processing.

If no file selection is specified for ZIP processing, then the PKZIP program will assume that there are no files to be added or updated, and will output an error message, whereas the PKUNZIP program will assume that ALL files in the Archive are to be processed.

&SYSUID

When specifying dataset names in commands or filename specifications within the command input stream, the reserved word "&SYSUID" may be used to represent the 1-7 character user name that the operating system supplies in the Address Space Control Block extension for the execution. ***PKZIP for MVS™*** will perform the substitution in the command string before continuing processing. By using this command notation, a generic set of commands can be set up to perform archiving operations for various users.

```
-ARCHIVE_DSN(&SYSUID.MY.ZIPS(SOURCE))
&SYSUID.MY.COBOL(*)
```

Summary of Available Commands

The commands (as summarized below) are available in both the PKZIP and PKUNZIP programs. Information specific to all commands will be found later in this chapter beginning with Command Details.

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
<dataset name>	Defines the name of a member that should be added to, updated in, or deleted from a compressed ZIP archive. Wildcards can be used to specify generic names.	●	●
–ACTION	**ADD** - Used to add files that are not already present in the ZIP archive. This is the default action for the PKZIP program (PKZIP default). **COPY** - Used to create a subset Archive from files contained in an existing Archive. **DELETE** - Specifies that selected files be deleted from the old ZIP archive. **EXTRACT** - Specifies that selected files be extracted from the ZIP archive. (PKUNZIP program default). **FRESHEN** - Specifies that selected files be updated in the old ZIP archive. **TEST** - Specifies that the ZIP archive files be tested for integrity. **UPDATE** - Used to update files that are already in the ZIP archive or to add files that are not already present in the ZIP archive. **VIEW** - Output details of the files selected from the ZIP archive to the SYSPRINT dataset.	●	●
–ARCHIVE_BLKSIZE	Specifies the block size for a new or updated ZIP archive.	●	

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–ARCHIVE_COMMENT	Allows a comment of up to 255 characters to be specified and saved in the archive central directory.	●	
–ARCHIVE_DATACLASS	Specifies the DF/SMS data class for a new or updated ZIP archive.	●	
–ARCHIVE_DIR_BLOCKS	Specifies the directory block amount for a new ZIP archive.	●	
–ARCHIVE_DSN	Specifies the archive to be read (and updated) by ZIP processing.	●	●
–ARCHIVE_DSORG	Specifies the dataset organization for a new or updated ZIP archive.	●	
–ARCHIVE_INFILE	Specifies the DDname that references a ZIP archive to be read in by the PKZIP program.	●	●
–ARCHIVE_LRECL	Specifies the logical record length for a new or updated ZIP archive.	●	
–ARCHIVE_MGMTCLASS	Specifies the DF/SMS management class to be used for a new or updated ZIP archive.	●	
–ARCHIVE_OUTFILE	Specifies a DD statement describing the archive to output to by ZIP processing.	●	
–ARCHIVE_RECFM	Specifies the record format of a new or updated ZIP archive.	●	
–ARCHIVE_SPACE_PRIMARY	Specify the number of allocation units in the primary extent of a new or updated ZIP archive.	●	
–ARCHIVE_SPACE_RLSE	Specifies whether free space should be released when the ZIP archive is de-allocated.	●	
–ARCHIVE_SPACE_SECONDARY	Specifies the number of allocation units in the secondary extent of a new or updated ZIP archive.	●	
–ARCHIVE_SPACE_TYPE	Specifies how space is to be allocated for a new or updated ZIP archive.	●	
–ARCHIVE_STORCLASS	Specifies the DF/SMS storage class for a new or updated ZIP archive.	●	
–ARCHIVE_TIMESTAMP	Specifies which Date/Time option to use in setting the timestamp of a created ZIP file.	●	
–ARCHIVE_UNIT	Specifies the generic unit for allocation of a new or updated ZIP file.	●	
–ARCHIVE_VOLUMES	Specifies the volume(s) for allocation of a new or updated ZIP archive.	●	
–ATTRIB_COMPATIBILITY	Governs the type of extended attributes that are stored in the Archive.	●	
–CALLMODE	Internal environmental interfacing command.	●	●
–CHECK_SYSIN_MEMBER	Verifies a command input stored in a PDS or PDSE member.	●	●
–COMPRESSION_LEVEL	Specifies speed and compression level when Zipping a file.	●	
–CRLF	Controls the use of record delimiters and an optional file terminator.	●	●

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–DATA_DELIMITER	Specifies the delimiter(s) to be used at the end of each text record of the file.	●	●
–DATA_STORAGE	Specifies the amount of cache memory used in ZIP processing.	●	●
–DATA_TYPE	Specifies that selected files for compression are binary or text. (Can be dynamically detected).	●	●
–DATATYPE_DETECT_DEPTH	Specifies the distance that a file is scanned before making a determination between binary or text.	●	●
–DATATYPE_DETECT_TABLE	Specifies the table of characters used to assess whether a byte is text or binary.	●	●
–DATATYPE_TEXT_PERCENT	Specifies the percentage of the sample that must meet the "text" criteria before it will be TEXT.	●	●
–DDNAME_PARMLIB	Specifies the DDname to use for command input (prior to SYSIN).	●	●
–DDNAME_SYSIN	Specifies the DDname to use for command input (unless –NOSYSIN is specified).	●	●
–DDNAME_SYSPRINT	Specifies the DDname to be used for ***PKZIP MVS™*** messages.	●	●
–DDNAME_ZPSORTIN	During –ACTION(VIEW) processing, SORT is called. This internal SORTIN DD is used.		●
–DDNAME_ZPSORTOUT	During –ACTION(VIEW) processing, SORT is called. This internal SORTOUT DD is used.		●
–ECHO	Specifies that a copy of ***PKZIP MVS™*** commands should be output to the message dataset.	●	●
–ENCRYTPION_METHOD	Specifies which encryption algorithm is to be employed.	●	
–EXCLUDE(dsname mask)	Specifies which files may be eliminated from being processed using a mask selection.	●	
–EXTRACT_PREVIEW	Specifies that a select number of records be processed for previewing the data.		●
–FILE_BUSY_WAITTIME	Specifies how long ***PKZIP MVS™*** should wait while continually retrying before it will terminate.	●	●
–FILE_EXTENSION	Specifies what to do with an extension.		●
–FILE_TERMINATOR	Specifies the character(s) to be stored (or recognized) at the end of the last record of a file.	●	●
–GDGALL_SUPPORT	Specifies whether all levels of a Generation Data Group (GDG) are to be retrieved and included in the archive.	●	
–GZIP	Specifies that the output archive will be created in GZIP format.	●	●
–GZIP_SUFFIX	Specifies the name to be used as the last level of the filename when there is no valid GZIP filename.	●	●
–HIERARCHY	Specifies that the full dataset component hierarchy should be used when converting a filename between ZIP archive format and MVS format.		●

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–INFILE	Specifies what file(s) to compress by identifying a DD statement.	●	
–INSERT_MEMBER	Used to add a member to an existing PDS.		●
–KEY_PROTECT_LEVEL	Strength of key protection for advanced encrypted archives.	●	
–LICENSE_HLQ	Specifies the high level qualifier to be used in locating the License Control Dataset.	●	●
–LMOD_SUPPORT	Sets –DATA_TYPE(BINARY), –SAVE_FILE_ATTRIBUTES, and –SAVE_LRECL commands on to allow simultaneous processing of load modules with text files in a PDS	●	●
–LOGGING_LEVEL	Specifies the level (or quantity) of messages output to SYSPRINT.	●	●
–MEMORY_MODEL	Specifies where file management control blocks are held.	●	
–MULTI_THREAD_LIMIT	Specifies where file management control blocks are held and the amount of storage than can be used for compression control tables.	●	●
–MULTI_THREAD_LIMIT	Specifies the number of subtasks to be used in compressing datasets.	●	●
–NOSYSIN	Specifies the SYSIN dataset is not opened for commands.	●	●
–ON_FILE_ACCESS_ERROR	Specifies the action to take when an access error has occurred.	●	●
–ON_FILE_IO_ERROR	Specifies the action to take when an I/O error has occurred.	●	●
–OUTDATAUSCL	This command is supported only in the ***PKZIP VSE***™ product.		●
–OUTFILE_BLKSIZE	Specifies the block size for a newly extracted dataset.		●
–OUTFILE_DATACLASS	Specifies the DF/SMS data class for a newly extracted dataset.		●
–OUTFILE_DD	Specifies what file(s) are to contain the extracted data by identifying a DD statement.		●
–OUTFILE_DIR_BLOCKS	Specifies the directory block amount for a newly extracted dataset.		●
–OUTFILE_DSNTYPE	Determines the type of output file to be created.		●
–OUTFILE_LRECL	Specifies the logical record length for a newly extracted dataset.		●
–OUTFILE_MGMTCLASS	Specifies the DF/SMS management class to be used for a newly extracted dataset.		●
–OUTFILE_OVERWRITE	Specifies overwrite of an existing file or member within a PDS.		●
–OUTFILE_PDS_ENQ	Specifies the level of disposition that will be used for a PDS or PDSE when processing an EXTRACT request.		●

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–OUTFILE_RECFM	Specifies the record format of a newly extracted dataset.		●
–OUTFILE_SPACE_PRIMARY	Specify the number of allocation units in the primary extent of a newly extracted dataset.		●
–OUTFILE_SPACE_RLSE	Specifies whether free space should be released when a newly extracted dataset is de-allocated.		●
–OUTFILE_SPACE_SECONDARY	Specify the number of allocation units in the secondary extent of a newly extracted dataset.		●
–OUTFILE_SPACE_TYPE	Specifies how space is to be allocated for a newly extracted dataset.		●
–OUTFILE_STORCLASS	Specifies the DF/SMS storage class for a newly extracted dataset.		●
–OUTFILE_UNIT	Specifies the generic unit for allocation of a newly extracted dataset.		●
–OUTFILE_VOLUMES	Specifies the volume(s) for allocation of a newly extracted dataset.		●
–OUTINDXUSCL	This command is supported only in the ***PKZIP VSE™*** product.		●
–OUTUSECLASS	This command is supported only in the ***PKZIP VSE™*** product.		●
–PAD_CHAR	Specifies the character to use to pad fixed length records when extracting.		●
–PAD_VSAM	Specifies that variable length records be padded using the character specified in –PAD_CHAR.		●
–PARMLIB_DSNAME_UNZIP	Specifies the name of the dataset containing the configuration specifications for UNZIP processing.	●	●
–PARMLIB_DSNAME_ZIP	Specifies the name of the dataset containing the configuration specifications for ZIP processing.	●	●
–PARMLIB_FILE_WAIT_MAX	If the specified –PARMLIB_DSNAME cannot be dynamically allocated, this is the amount of time to wait before terminating.	●	●
–PARMLIB_FILE_WAIT_TIMER	If the specified –PARMLIB_DSNAME cannot be dynamically allocated, this is the amount of time to wait before retrying (up to –PARMLIB_FILE_WAIT_MAX.	●	●
–PASSWORD	Specifies a password to encrypt/decrypt ZIP archive files.	●	●
–PATCH_REPORT	Specifies that a report of all patches be produced. See –ACTION.	●	●
–PATH	Specifies that only the last component of the dataset component hierarchy should be used when converting a filename between MVS format and ZIP archive format.	●	
–PKSUPPRC	A default command that allows the return code to be suppressed.	●	●
–PRESERVE_CMD_SPACES	Preserves or removes blanks preceding the "|".	●	●

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–PROCESS_ALIAS	Specifies whether the alias entries for selected PDS members are to be used.	●	●
–RECALL_TO_ZIP	Specifies whether DFHSM recall of datasets should occur.	●	
–RECURSE_LEVELS	Specifies whether or not data components beyond those specified should be used in matching with your selection.	●	
–SAVE_FILE_ATTRIBUTES	Specifies where file attributes should be stored for datasets in the zip archive; in the Central Directory only, the Local Directory, both directories, or neither directory.	●	●
–SAVE_LRECL	Compress/ Decompress a binary file with variable record lengths.	●	●
–SELECT_CATALOGED_ALIAS	Specifies whether aliases are to be supported at time of zipping.	●	
–SELECT_FROM_PDS	Specifies a PDS dataset from which ***PKZIP MVS***™ can obtain members to match user selection parameters that do not match any other dataset.	●	●
–SELECT_TAPE	Specifies whether tape files are to be retrieved and included in the archive.	●	
–SET_ERROR_RC	Specifies a firm return code to be passed to the system when an error has been detected.	●	●
–SHOW_SETTINGS	Displays all current command settings.	●	●
–SIMULATE	Simulates file selection processes, but does not perform actual data manipulation for the files selected.	●	●
–SNAP_SYSOUT_CLASS	Specifies the SYSOUT class to be used for SNAP dumps (reserved for future use).	●	●
–STAGE_TAPE_ON_DISK	Specifies input from a sequential device be stored in a temporary dataset.	●	●
–STRIP_CHAR	Specifies an ending character to be removed from the end of each record before it is zipped.	●	
–SUPPRESS_DYNALLOC_MSGS	Specifies that the dynamic allocation messages in job log be suppressed.	●	●
–SYSPRINT_DCB	Specifies the DCB record format to be used for SYSPRINT listings.	●	
–SYSPRINT_SYSOUT_CLASS	Specifies the JES SYSOUT class that will be used for the SYSPRINT listing.	●	●
–TAPETODISK	Specifies that cartridge based archives should be copied to a temporary disk file for processing.	●	●
–TASKS	Specifies the number of subtasks used to compress datasets.	●	
–TEMP_BLKSIZE	Specifies the temporary block size of a temporary ***PKZIP MVS***™ dataset.	●	●
–TEMP_DATACLASS	Specifies the DF/SMS data class to be used for a temporary ZIP dataset.	●	●

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–TEMP_MGMTCLASS	Specifies the DF/SMS management class to be used for a temporary file allocation.	●	●
–TEMP_RECFM	Specifies the record format for a temporary ZIP dataset.	●	●
–TEMP_SPACE_PRIMARY	Specifies the number of space units to be used in the primary partition of a temporary ZIP dataset.	●	●
–TEMP_SPACE_SECONDARY	Specifies the number of space units to be used in the secondary partition of a temporary ZIP dataset.	●	●
–TEMP_SPACE_TYPE	Specifies how space is to be allocated for a temporary ZIP dataset.	●	●
–TEMP_STORCLASS	Specifies the DF/SMS storage class to be used for a temporary file allocation.	●	●
–TEMP_UNIT	Specifies the unit to be used for allocation of a temporary ZIP dataset.	●	●
–TEMP_VOLUMES	Specifies the volume onto which a temporary ZIP dataset should be placed.	●	●
–TRACE_TABLE_SIZE	Specifies the size of the internal trace table.	●	●
–TRANSLATE_TABLE_DATA	Specifies which translation table to use when converting character sets of text files.	●	●
–TRANSLATE_TABLE_FILEINFO	Specifies a translation table to be used with file information such as comments, file names, and control information of a ZIP archive.	●	●
–TRANSLATION_MODE	(Reserved for future use).	●	●
–UNZIPPED_DSN	Specifies a different high-level qualifier for an extracted dataset.		●
–VSAM	Specifies whether VSAM files should be used or ignored when selecting files for compression and using wildcards.	●	
–VSAM_ACCOUNT	Specifies the accounting information to be provided to Access Methods Services during a DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_ATTEMPTS	Specifies the number of password attempts that are permitted to Access Methods Services during a DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_AUTH_EP	Supplies the entry point of a user security verification routine to Access Methods Services during a DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_AUTH_STRING	Supplies a string of information to be passed to your security verification routine to Access Methods Services during a DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_BUFFERSPACE	Specifies the BUFFERSPACE parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–VSAM_CATALOG	Specifies the CATALOG parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_CISIZE	Specifies the CONTROLINTERVALSIZE parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_CLUSTER_TYPE	Specifies the file type to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_CODE	Supplies a code name for the cluster or component to Access Methods Services during a DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_CONTROLPW	Specifies the CONTROLPW parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_DATA_CISIZE	Specifies the CONTROLINTERVALSIZE parameter to the data component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_DATA_EXCEPTIONEXIT	Specifies the EXCEPTIONEXIT parameter to the data component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_DATA_FILE	Specifies the FILE parameter to the data component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_DATA_NAME	Specifies the NAME parameter to the data component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	
–VSAM_DATA_ORDERED	Specifies the ORDERED parameter to the data component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_DATA_PRIMARY	Specifies the primary space allocation value to the data component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_DATA_SECONDARY	Specifies the secondary space allocation value to the data component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_DATA_SPACE_TYPE	Specifies the space allocation type parameter to the data component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_DATA_VOLUMES	Specifies the VOLUMES parameter to the data component of an IDCAMS DEFINE CLUSTER command, used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_DATACLASS	Specifies the DF/SMS data class to be used for the creation of a new (or update of an existing) VSAM-defined ZIP Archive.	●	●

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–VSAM_DUPLICATE_ERROR	Specifies the action to be taken on realization of a duplicate key when creating a new extracted VSAM dataset.		●
–VSAM_ERASE	Specifies the ERASE parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_EXCEPTIONEXIT	Specifies the EXCEPTIONEXIT parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_FILE	Specifies the FILE parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	
–VSAM_FOR	Specifies the FOR parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	
–VSAM_FREESPACE_CA	Specifies the FREESPACE parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_FREESPACE_CI	Specifies the FREESPACE parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_IMBED	Specifies the IMBED parameter of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_ATTEMPTS	Specifies the number of password attempts that are permitted to Access Methods Services during a DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_AUTH_EP	Supplies the entry point of a user security verification routine to Access Methods Services during a DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_AUTH_STRING	Supplies a string of information to be passed to your security verification routine to Access Methods Services during a DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_CISIZE	Specifies the CONTROLINTERVALSIZE parameter to the INDEX component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_CODE	Supplies a code name for the cluster or component to Access Methods Services during a DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_CONTROLPW	Specifies the CONTROLPW parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_EXCEPTIONEXIT	Specifies the EXCEPTIONEXIT parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–VSAM_INDEX_FILE	Specifies the FILE parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_MASTERPW	Specifies the MASTERPW parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_NAME	Specifies the NAME parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_ORDERED	Specifies the ORDERED parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_PRIMARY	Specifies the primary space allocation parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_READPW	Specifies the READPW parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_SECONDARY	Specifies the secondary space allocation parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_SPACE_TYPE	Specifies the space allocation type parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_UPDATEPW	Specifies the UPDATEPW parameter to the index component of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_INDEX_VOLUMES	Specifies the VOLUMES parameter to the index component of an IDCAMS DEFINE CLUSTER command, used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_KEYS	Specifies the KEYS parameter for an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_MASTERPW	Specifies the MASTERPW parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_MGMTCLASS	Specifies the DF/SMS management class to be used for the creation of a new (or update of an existing) VSAM-defined ZIP Archive.		●
–VSAM_MODEL	Specifies the MODEL parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_ORDERED	Specifies the ORDERED parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–VSAM_OWNER	Specifies the OWNER parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	
–VSAM_READPW	Specifies the READPW parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_RECORDSIZE	Specifies the RECORDSIZE parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_RECOVERY_OPT	Specifies the SPEED or RECOVERY parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_REPLICATE	Specifies the REPLICATE parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_REUSE	Specifies the REUSE|NOREUSE parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_SHAREOPTIONS	Specifies the SHAREOPTIONS parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_SPACE_PRIMARY	Specifies the number of allocation units to be allocated in the primary extent of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_SPACE_SECONDARY	Specifies the number of allocation units to be allocated in the secondary extent of an IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_SPACE_TYPE	Specifies the type of allocation units to be allocated in the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	
–VSAM_SPANNED	Specifies the SPANNED|NONSPANNED parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_STORCLASS	Specifies the DF/SMS storage class to be used for the creation of a new (or update of an existing) VSAM-defined ZIP Archive.		●
–VSAM_TO	Specifies the TO parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.	●	●
–VSAM_UPDATEPW	Specifies the UPDATEPW parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an existing) VSAM-defined ZIP Archive.		●
–VSAM_USECLASS	This command is supported only in the ***PKZIP VSE™*** product.		
–VSAM_WRITECHECK	Specifies the WRITECHECK|NOWRITECHECK parameter to the IDCAMS DEFINE CLUSTER command used to create a new (or update an	●	●

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
	existing) VSAM-defined ZIP Archive.		

COMMAND	DESCRIPTION	PKZIP	PKUNZIP
–ZIPPED_DSN	Specifies what parameters to use in converting MVS file names to ZIP file names.	●	
–ZIPPED_DSN_SEPARATOR	Specifies what separator to use in the new ZIP archive name.	●	

Command Details

Descriptions of ***PKZIP for MVS™*** commands are shown below in alphabetic sequence. If applicable s4ynonyms for each command are listed directly below the command.

<dataset name>

The <dataset name> is an individual name or a file mask of files that are to be used in the ZIP or UNZIP process. The specification may represent one or more files when either wildcard masks are used, or RECURSE_LEVELS is specified.

> **Note: This command does not have a "–" prefixing it.**

Pathnames may be specified in the <dataset name> and may be either in MVS format (MYFILES.PROJECT.DATA) where periods separate the qualifiers or in UNIX format that use slashes (MYFILES/PROJECT/DATA). ***PKZIP for MVS™*** stores the <dataset name> in the latter format to provide cross-platform compatibility but accepts references to <dataset name> in MVS format.

> **Note: When standard ZIP Archives are requested, a filename may be of mixed case. When GZIP is requested, all characters in the filename will be lower case according to GZIP specifications.**

FORMATTING:

For individual datasets or PDS names the <dataset name> entry consists of:

```
dataset level[.dataset level][.dataset level]….
```

For example: mydata.testing.temp.

For PDS members, the <dataset name> entry consists of:

```
dataset level[.dataset level][.dataset level] ...
     (member1[,member2][,member3]…)
```

For example: mydata.testing.temp(firstrun,secondrun).

When a single dataset level is specified either as a dataset or a PDS member, and if:

–SELECT_FROM_PDS is present, the associated PDS is identified.

–SELECT_FROM_PDS is not present, then the single level will be identified as a high level qualifier.

Wildcard - A method to specify several datasets by using one name. Special characters (?, *, and **) replace part or all of the name and operate as "wild cards."

Wildcards may not be used in the highest dataset level of the dataset name.

The more general the wildcard specifications, the longer the file search. To save time, be as specific as possible in selecting dataset names.

A question mark (?) - is used to allow all occurrences of letters or characters in that position within a dataset level.

MBS.?ABC would identify all datasets that have one character before ABC in its dataset level. For example:

MBS.1ABC

MBS.2ABC

MBS.MABC

MBS.??ABC would include datasets that have two wildcard characters before ABC in the dataset level. For example:

MBS.10ABC

MBS.XXABC

MBS.1JABC

***** - is used to allow all occurrences of names or partial names in that position. It may represent zero or more characters within the level.

JEH.*.SUB represents all datasets of any second level and a third level of SUB datasets. For example:

JEH.BVC.SUB

JEH.TRIAL.SUB

JEH.UNVTEST.SUB

JEH.A*.SUB represents all datasets with a third level of .SUB and all second level beginning with A. For example:

JEH.ABC.SUB

JEH.AQZAR.SUB

JEH.ATEST.SUB

BOOT.* represents all datasets with a first component of BOOT plus any of its second levels. It does not represent datasets with more than one level (see ** for more than one). For example:

BOOT.MINE

BOOT.DATA

BOOT.TESTING

but not

BOOT.MINE.SOURCE

JEH.*.D* represents all files within JEH with D beginning its third level. For example:

JEH.OWN.DATA

JEH.SOURCE.DELIM

JEH.BAKER.DEMO

****** - is used to allow all occurrences of ONE or the next TWO datasets levels.

ABC.** represents all datasets beginning with ABC and the next one or two levels, if present. For example:

ABC.GROUP.TEST

ABC.GROUP

ABC.MINE

ABC.**.DATA represents datasets with the first level of ABC followed by one or two level(s) and ending with DATA as the last level. For example:

ABC.GROUP.BASIC.DATA

ABC.GROUP.DATA

ABC.MINE.DATA

MS-DOS and UNIX file formats

Dataset names are supported in MS-DOS and UNIX formats to delete or view entries. For all other operations, dataset names should be in the MVS format.

For UNIX or MS-DOS formatting:

[pathname][/pathname]…[/pathname][/filename]

For MS-DOS formatting:

[pathname][\pathname]…[\pathname][\filename]

Command Icon Legend

The following legend is used to identfy icons that may be associated with a given command. These icons provide platform information, command compatibility, and a ☠ icon indicates that you should proceed with extreme caution and double check that the information provided works with your platform. It is important that you double check a command before using it.

Icons	Description
☺	This icon specifies what platforms use this command.
☹	This command is not compatible with UNIX, MVS, OS/400, and Windows.
☠	This icon is a **warning** and it instructs you to read the information and proceed with **caution**.

–ACTION

Synonyms Include: –ADD, –COPY, –DELETE,
–EXTRACT, –FRESHEN, –PATCH_REPORT, –TEST, –UPDATE,
–VIEW

The –ACTION command is used to add, copy, delete, extract, freshen, update, or view files in a ZIP archive. It may also be used to view a patch report.

–ACTION(ADD|COPY|DELETE|EXTRACT|FRESHEN|PATCH_REPORT|
TEST|UPDATE|VIEW)

ADD - Specifies the addition of a file(s) to a ZIP archive using the method as specified in –COMPRESSION_METHOD. If a file already exists in the archive with the same name, the addition will be disallowed and an UPDATE modifier will be required.

Use –ARCHIVE_DSN or a combination of –ARCHIVE_INFILE and –ARCHIVE_OUTFILE along with the –ACTION(ADD) to create the new ZIP archive.

The ADD command will force the creation of a new ZIP archive.

ADD is the default action for the PKZIP program.

COPY - Specifies that designated files (all by default) are to be copied from one archive to another when running program PKZIP. Dataset name selections are accomplished the same as they are with –ACTION(DELETE) defined previously. When no names are specified, all files within the input archive will be copied to the target. No action will be taken if the target archive is the same as the source archive.

Use of –ARCHIVE_DSN in conjunction with –COPY will cause an implicit deletion of all files not selected from the designated archive. This can be a more efficient method of deleting files from an Archive, instead of listing them all with –DELETE. PKZIP will not allow an implicit deletion of all files within an archive when using–COPY.

When –ARCHIVE_INFILE is used in conjunction with –COPY, PKZIP will allow the creation of an empty target archive when using –COPY and none of the requested files match the input archive.

DELETE - Specifies that the file(s) selected by the <dataset name> command be deleted from an existing ZIP archive. This action will result in the creation of a new archive, minus the deleted files.

Use –ARCHIVE_DSN (or a combination of –ARCHIVE_INFILE and –ARCHIVE_OUTFILE) along with the –ACTION(DELETE) to create the new ZIP archive.

The DELETE command will force the creation of a new ZIP archive minus the deleted files.

EXTRACT - Specifies that items or files are looked for within the archive, brought out, and put into an MVS dataset. EXTRACT is the default action for the PKUNZIP program.

FRESHEN - Specifies that a file(s) already existing within an archive is to be replaced by a different file with the same name. Note that timestamp verification does not occur, so it is possible to replace a current file with one that is older.

PATCH_REPORT - When gathering information for problem analysis, ***PKZIP for MVS™*** Technical Support may request the output from an execution with –PATCH_REPORT. The report output is sent to the designated –DDNAME_SYSPRINT standard output. No other commands are required.

This is normally executed in batch, although a Foreground report can be generated with the ISPF panels:

Note: This command may be used under either PGM=PKZIP or PGM=PKUNZIP. No archive actions will be performed when this command action is selected.

TEST - Specifies that the ZIP archive files be tested for integrity.

This command will perform the same functions as an –ACTION(EXTRACT) command without actually extracting data or producing a decompressed file. The stored CRC is checked in this process and a confirmation message will occur in the SYSPRINT dataset for each valid file.

Use –ARCHIVE_DSN or –ARCHIVE_INFILE with this command to specify the ZIP archive to be validated.

UPDATE - Specifies the update or addition of a file(s) to an existing ZIP archive.

VIEW - Specifies that information about selected files be displayed in SYSPRINT. The VIEW command may be used with or without parameters. All parameter fields are optional, but if specified, must be specified in the following order:

VIEW[level][sort][REVERSE][COMMENT]

Level - This parameter specifies the amount and format of the information to be displayed.

- **Null** - If no level is specified, a standard report of one line per file (wrap lines may be inserted for the file name or comment) will be displayed with columnar headings for the field values.
- **BRIEF** - Provides a minimum of information about the files selected for display.
- **DETAIL** - Provides a full set of technical details about the files selected for display.

Sort - Determines the presentation sequence of information in the output report.

- **NAME** - Sort by filename only.
- **DATE** - Sort by date only.
- **LENGTH** - Sort by length of the uncompressed file only.
- **OFFSET** - Sort by order of occurrence within the ZIP archive (first in, first out). This is the default sort sequence.
- **PERCENT** - Sort by compression percentage, only.
- **SIZE** - See Length.

REVERSE - This optional switch will display the files in the opposite order as specified. For example, a NAME sort will display the files in ascending order, but if NAMEREVERSE is specified, then the files will be in descending order according to the file name.

COMMENT - This optional parameter specifies that if an internal comment is in the Archive Directory for the listed file(s), a separate line will list the comment information for its associated file. This is different than the –ARCHIVE_COMMENT, which applies to the entire archive.

Following are valid –ACTION(VIEW) options:

VIEWBRIEF	VIEWDETAILLENGTHREVERSECOMMENT
VIEWBRIEFCOMMENT	VIEWDETAILNAME
VIEWBRIEFDATE	VIEWDETAILNAMECOMMENT
VIEWBRIEFDATECOMMENT	VIEWDETAILNAMEREVERSE
VIEWBRIEFDATEREVERSE	VIEWDETAILNAMEREVERSECOMMENT
VIEWBRIEFDATEREVERSECOMMENT	VIEWDETAILOFFSET
VIEWBRIEFLENGTH	VIEWDETAILOFFSETCOMMENT
VIEWBRIEFLENGTHCOMMENT	VIEWDETAILOFFSETREVERSE
VIEWBRIEFLENGTHREVERSE	VIEWDETAILOFFSETREVERSECOMMENT
VIEWBRIEFLENGTHREVERSECOMMENT	VIEWDETAILPERCENT
VIEWBRIEFNAME	VIEWDETAILPERCENTCOMMENT

VIEWBRIEFNAMECOMMENT	VIEWDETAILPERCENTREVERSE
VIEWBRIEFNAMEREVERSE	VIEWDETAILPERCENTREVERSECOMMENT
VIEWBRIEFNAMEREVERSECOMMENT	VIEWDETAILREVERSE
VIEWBRIEFOFFSET	VIEWDETAILREVERSECOMMENT
VIEWBRIEFOFFSETCOMMENT	VIEWDETAILSIZE
VIEWBRIEFOFFSETREVERSE	VIEWDETAILSIZECOMMENT
VIEWBRIEFOFFSETREVERSECOMMENT	VIEWDETAILSIZEREVERSE
VIEWBRIEFPERCENT	VIEWDETAILSIZEREVERSECOMMENT
VIEWBRIEFPERCENTCOMMENT	VIEWLENGTH
VIEWBRIEFPERCENTREVERSE	VIEWLENGTHCOMMENT
VIEWBRIEFPERCENTREVERSECOMMENT	VIEWLENGTHREVERSE
VIEWBRIEFREVERSE	VIEWLENGTHREVERSECOMMENT
VIEWBRIEFREVERSECOMMENT	VIEWNAME
VIEWBRIEFSIZE	VIEWNAMECOMMENT
VIEWBRIEFSIZECOMMENT	VIEWNAMEREVERSE
VIEWBRIEFSIZEREVERSE	VIEWNAMEREVERSECOMMENT
VIEWBRIEFSIZEREVERSECOMMENT	VIEWOFFSET
VIEWCOMMENT	VIEWOFFSETCOMMENT
VIEWDATE	VIEWOFFSETREVERSE
VIEWDATECOMMENT	VIEWOFFSETREVERSECOMMENT
VIEWDATEREVERSE	VIEWPERCENT
VIEWDATEREVERSECOMMENT	VIEWPERCENTCOMMENT
VIEWDETAIL	VIEWPERCENTREVERSE
VIEWDETAILCOMMENT	VIEWPERCENTREVERSECOMMENT
VIEWDETAILDATE	VIEWREVERSE
VIEWDETAILDATECOMMENT	VIEWREVERSECOMMENT
VIEWDETAILDATEREVERSE	VIEWSIZE
VIEWDETAILDATEREVERSECOMMENT	VIEWSIZECOMMENT
VIEWDETAILLENGTH	VIEWSIZEREVERSE
VIEWDETAILLENGTHCOMMENT	VIEWSIZEREVERSECOMMENT
VIEWDETAILLENGTHREVERSE	

–ARCHIVE_BLKSIZE

Synonyms Include: –ARCHBLKSIZ

For a new or updated ZIP archive, the block size may be specified using the –ARCHIVE_BLKSIZE command. The default is to attempt half-track blocking, for example, 27998 bytes, on a 3390 DASD device unless –ARCHIVE_LRECL is specified using logical record lengths. The default is not used when –ARCHIVE_DATACLASS is specified and DF/SMS is in control of dataset allocation on the system.

–ARCHIVE_BLKSIZE(<block size>)

block size - This is the size of block for the new or updated ZIP archive.

Note: A large block size should be specified for best ZIP performance.

Block size of 0:

If using a PDS or sequential archive, and a block size of 0 is specified, the program will determine the block size.

If using record formats that are undefined, by default or by –ARCHIVE_RECFM(U) command, and a block size of 0 is specified, the system may not set a block size. An error will occur when the archive is processed, such as, `IEC141I 013-34` abend.

–ARCHIVE_COMMENT

Synonyms Include: N/A

This command allows a comment of up to 255 characters to be specified and saved in the archive central directory.

–ARCHIVE_COMMENT(<comment>)

comment - A free form descriptive field that may be up to 255 characters in length and may contain lower-case letters.

```
-ARCHIVE_COMMENT(This is a sample of a long command input value, and a hyphen
   illustrates the use of the continuation character for a lon-
…..g command.)
   The hyphen causes a concatenation without blanks.
```

–ARCHIVE_DATACLASS

Synonyms Include: –ARCHDCLASS

For a new or updated ZIP archive, the SMS data class may be specified using the –ARCHIVE_DATACLASS command. If the command is not specified, no data class is used in the allocation request.

Allocation of files in a SMS environment is controlled by the installation through Automatic Class Selection routines as defined by the local storage administrator. Control cards specifying SMS classes and/or volume selection may be ignored by the system when performing allocations. Check with the systems administrator for proper designations of these values.

–ARCHIVE_DATACLASS(<data class>)

data class - Names the SMS data class where the updated or new archive is to reside. There is an 8-character limit.

–ARCHIVE_DIR_BLOCKS

Synonyms Include: –ARCHDIRBLKS, –ARCHIVE_DIRBLKS

For a new ZIP archive, the number of directory blocks may be specified using the –ARCHIVE_DIR_BLOCKS command. The default of 56 is not used with –ARCHIVE_DATACLASS.

Use –ARCHIVE_DIR_BLOCKS in conjunction with an –ARCHIVE_DSN when creating a new PDS.

–ARCHIVE_DIR_BLOCKS(<dir blocks>)

dir blocks - This indicates the number of directory blocks for the new ZIP archive. The default allocation is 56 blocks.

–ARCHIVE_DSN

Synonyms Include: –ARCHIVE, –ARCHIVE_DSNAME

When used in PKZIP program processing:

The –ARCHIVE_DSN command specifies the archive name to be read in and updated by ***PKZIP for MVS™***. Either this command or the –ARCHIVE_INDD command must be used to identify an archive. –ARCHIVE_INDD does not allow updating and is used in conjunction with –ARCHIVE_OUTDD. There is no default.

–ARCHIVE_DSN(<archname>)

archname - This is the complete archive dataset name of the ZIP archive. If the archive is a PDS archive, the member name must be included here.

If **archname** exists:

PKZIP for MVS™ will perform a SYSTEM ENQUE to lock out other users from accessing the archive.

PKZIP for MVS™ will create a temporary file(s) of compressed data as processing occurs. Once complete, the old archive is deleted and the temp file takes on the archive name, creating an updated archive.

The updated archive will have allocation attributes from –ARCH* commands or their defaults instead of the previous archive's allocation.

> **Note: The temporary file(s) may require as large an allocation as the archive itself. Use the –TEMP* commands to specify sufficient allocation.**

If the archive came from another platform, the created dataset must be created on MVS as sequential or as a PDS member with type U, F, or FB records. Generate this dataset with a block size of at least 4000 bytes for best processing.

PKZIP for MVS™ will create the archive with the <archive name>.

If this is to be a first member of a PDS, use –ARCHIVE_DIR_BLOCKS to specify the allocation of directory blocks or use the default.

When used in PKUNZIP program processing:

The –ARCHIVE_DSN command specifies the archive name to be read in or viewed by the PKUNZIP program.

> **Note: Either this command or the –ARCHIVE_INDD command must be used to identify an archive. There is no default.**

–ARCHIVE_DSN(<archname>)

archname - This is the complete dataset name of the ZIP archive.

PKZIP for MVS™ will perform a SYSTEM ENQUE to lock out other users from using the archive.

–ARCHIVE_DSORG

Synonyms Include: –ARCHDSORG

For a new or updated ZIP archive, the dataset organization is specified using the –ARCHIVE_DSORG command. The command may specify one of four organizations with Sequential the default. Note, with the exception of VSAM files ***PKZIP for MVS™*** can determine the dataset organization by the dataset name in the
–ARCHIVE_DSN command.

–ARCHIVE_DSORG(PO|PE|PS|VS)

PO - Partitioned DataSet archive.

PE - Partitioned DataSetenhanced archive.

PS - Physical Sequential archive.

VS - Virtual Storage Aaccess Method archive.

Note: The program can determine the organization of the archive by the dataset name with the exception of VSAM files.

–ARCHIVE_INFILE

Synonyms Include: –ARCHINDD, –ARCHIFILE, –ARCHINFILE,–ARCHIVE_INDD, –ARCHIVE_IFILE

The –ARCHIVE_INFILE command specifies a DD statement that describes a ZIP archive to be read in for processing. Use this command when the archive is not to be updated and the processed file is to be written to another destination using –ARCHIVE_OUTFILE. Also use this command when processing tapes and GDG's. Do not use this command in conjunction with the –ARCHIVE_DSN command.

```
–ARCHIVE_INFILE(<DDname>)
```

DDname - This is the DD statement in the JCL that identifies the ARCHIVE to be read.

The same <DDname> may not be used for –ARCHIVE_OUTFILE.

–ARCHIVE_LRECL

Synonyms Include: –ARCHLRL

For a new or updated ZIP archive, the logical record length is specified using the –ARCHIVE_LRECL command. If –ARCHIVE_RECFM(U) is specified for sequential archives then a default record length of 0 is established otherwise the block size is used . Note that the command –ARCHIVE_DATACLASS overrides this default.

```
–ARCHIVE_LRECL(<lreclength>)
```

lreclength - This is the logical record length for the new or updated ZIP archive.

–ARCHIVE_MGMTCLASS

Synonyms Include: –ARCHMCLASS

For new file allocation when doing PKUNZIP processing these classes will be passed to SMS when dataset allocation occurs.

```
–ARCHIVE_MGMTCLASS(<SMS Management Class>)
```

See IBM's DF/SMS manuals for further information about this parameter.

–ARCHIVE_OUTFILE

Synonyms Include: –ARCHIVE_OUTDD, –ARCHIVE_OFILE, –ARCHOUTDD, –ARCHOFILE, –ARCHOUTFILE

The –ARCHIVE_OUTFILE command specifies a DD statement that points to a ZIP archive to be written. Use this command when the input archive is not to be updated with new information. This command is mainly used when processing tapes and GDG's. Do not use this command in conjunction with the –ARCHIVE_DSN command.

```
–ARCHIVE_OUTFILE(<DDname>)
```

DDname - This is the DD statement in the JCL that identifies the ARCHIVE to write. It must not be the same as used for –ARCHIVE_INFILE.

If the archive is updated, the JCL parameter DISP=MOD should not be used to extend the archive. DISP=OLD should be used instead to allow the archive to be overwritten.

If the archive is not updated, then the input archive will be copied to the <DDname> archive. The <DDname> attributes in the JCL are used to define the output archive. Any –ARCH* commands will be ignored.

In the event of an error occuring during ZIP processing and the process does not complete, the output dataset within the archive should not be used. The status of the dataset is determined once the process completes and therefore will not be determined if an error is encountered.

–ARCHIVE_RECFM

Synonyms Include: –ARCHTYPE

For a new or updated ZIP archive, the record format may be specified using the –ARCHIVE_RECFM command. The record specification may be one of four types with U (Undefined) as the default.

–ARCHIVE_RECFM(U|F|FB|FBS)

U - Undefined records (default) (note also that this default is ignored if an associated SMS command of –ARCHIVE_DATACLASS is used).

F - Fixed records.

FB - Fixed-Block records.

FBS - Fixed-Block Standard records.

An undefined specification (U) will cause any –ARCHIVE_LRECL specifications to be ignored. Similarly, an unblocked file specification will cause –ARCHIVE_BLKSIZE to be ignored.

–ARCHIVE_SPACE_PRIMARY

Synonyms Include: –ARCHPRIMARY

For a new or updated ZIP archive, the number of allocation units in the primary extent is specified using the –ARCHIVE_SPACE_PRIMARY command.

The default is not used if –ARCHIVE_DATACLASS is specified.

–ARCHIVE_SPACE_PRIMARY(<allocation units>)

allocation units - This is an 8-character field specifying the number of allocation units for the primary extent of the new or updated ZIP archive.

00000010 - Ten (cylinders) is the default.

Allocation units are automatically released for a sequential archive.

–ARCHIVE_SPACE_RLSE

Synonyms Include: –ARCHIVE_RLSE, –ARCHIVE_RELEASE, –ARCHIVE_SPACE_RELEASE, –ARCHRLSE, –NOARCHRLSE, –ARCHNORLSE

This command specifies whether free space should be released when a ZIP archive is deallocated.

–ARCHIVE_SPACE_RLSE(Y|N)

Y - YES - The deallocated free space is released following compression. This is the default action taken for sequential datasets.

N - NO - The deallocated free space is not released following compression. This is the default action taken for partitioned datasets.

–ARCHIVE_SPACE_SECONDARY

Synonyms Include: –ARCHSECONDARY

For a new or updated ZIP archive, the number of allocation units in the secondary extent is specified using the –ARCHIVE_SPACE_SECONDARY command. If specified, the data unit number must not be 0.

The default is not used if –ARCHIVE_DATACLASS is specified.

allocation units - This is an 8-character field specifying the number of allocation units for the secondary extent of the new or updated ZIP archive.

00000010 - Ten (cylinders) is the default.

–ARCHIVE_SPACE_TYPE

Synonyms Include: –ARCHSPACE

For a new or updated ZIP archive, the type of allocation units may be specified using the –ARCHIVE_SPACE_TYPE command. Note the default is not used when –ARCHIVE_DATACLASS is specified.

```
–ARCHIVE_SPACE_TYPE(<TRK|CYL|BLK|MB|KB>)
```

TRK - (also TRKS and TRACKS) Allocation by tracks.

CYL - (also CYLS and CYLINDERS) Allocation by cylinders.

BLK - (also BLKS and BLOCKS) Allocation by blocks (Note that the block size is specified in the –ARCHIVE_BLKSIZE command.

KB - (also KILOBYTES) Allocation by Kilobytes for a VSAM archive.

MB - (also MEGABYTES) Allocation by Megabytes for a VSAM archive.

> **VSAM Note: Both the primary and secondary extents are allocated at 100 allocation units unless changed by the –VSAM_SPACE_PRIMARY or the –VSAM_SPACE_SECONDARY commands.**

This command specification can be overridden at the data level by the –VSAM_DATA_SPACE_TYPE command. At the data level, the corresponding cluster information is not recognized.

–ARCHIVE_STORCLASS

Synonyms Include: –ARCHSCLASS

For a new or updated ZIP archive, the DF/SMS storage class may be specified using the –ARCHIVE_STORCLASS command. If the command is not specified no storage class is used.

```
–ARCHIVE_STORCLASS(<storclass>)
```

storclass - These are the names of the DF/SMS storage class where the updated or new archive is to reside. There is an 8-character limit.

For new ZIP archives that are members of a PDS, the ***PKZIP for MVS™*** DF/SMS command should specify the PDS class and the non-DF/SMS command should specify the PDS volume or unit of the allocation.

–ARCHIVE_TIMESTAMP

Synonyms Include: –TIMESTAMP

This command specifies the source of the date and time for a compressed file. The default is the LOCAL time, as set on the system.

–ARCHIVE_TIMESTAMP(CREATE|CREATEGMT|CREATEUTC|GMT|LOCAL|UTC)

CREATE - Specifies the creation date of the MVS dataset with time of 00:00:00. This is because standard MVS systems retain the dataset's creation date but do not retain the time of creation. If this creation date does not exist, the LOCAL time is used. Members of a PDS will have the timestamp associated with the dataset, not with the individual members.

CREATEGMT - Specifies the creation date of the MVS dataset with a time of 00:00:00 as in CREATE. Except if the creation date does not exist, the UTC option is used.

CREATEUTC - Specifies the creation date of the MVS dataset with a time of 00:00:00 as in CREATE. Except if the creation date does not exist, the UTC option is used.

GMT - Specifies the Greenwich Mean Time as set on the system. Time zones are not specified here; therefore, it is the same time, world-wide. The time is captured at the time ZIP processing begins.

LOCAL - Specifies the LOCAL time as set with the system. The LOCAL time is based on the UTC time with any adjustments made for time zones.

UTC - Specifies the Greenwich Mean Time as set on the system. Time zones are not specified here; therefore, it is the same time, world-wide. The time is captured at the time ZIP processing begins.

The time captured for the archive is the point at which ZIP processing begins and is the same for all files of that archive.

–ARCHIVE_UNIT

Synonyms Include: –ARCHUNIT

For new or updated ZIP file allocation, the generic units for the archive can be specified using the –ARCHIVE_UNIT command. The default, should a unit be required, is the installation default, typically SYSDA.

–ARCHIVE_UNIT(unitname|SYSDA)

unitname - This is an 8-character field specifying the name of the generic unit to which the archive is to be allocated.

SYSDA - This is the default specification.

For new ZIP archives that are members of a PDS, the ***PKZIP for MVS***™ DF/SMS command should specify the PDS class and the non-DF/SMS command should specify the PDS volume or unit of the allocation.

–ARCHIVE_VOLUMES

Synonyms Include: –ARCHVOL

For a new or updated ZIP archive allocation, the volume(s) is specified using the –ARCHIVE_VOLUMES command.

–ARCHIVE_VOLUMES(<volname>[<volname> <volname>…])

volname - This is a 217-byte field specifying the name of the volume(s) onto which the new or updated ZIP archive is allocated. There may be up to 31 volume names specified with this command.

For an archive that is a new member of a new PDS, the first <volname> will only be used.

For a VSAM archive, the volumes are specified at the Cluster Level.

–ATTRIB_COMPATIBILITY

Synonyms Include: –ATTRCOMPAT, –ATTRIB_COMPAT, –ATTRIBUTE_COMPATIBILITY

This parameter governs the type of extended attributes that are stored in the Archive. ***PKZIP for MVS™*** Version 5.5 provides compatible attributes with ***PKZIP for VSE™*** Version 5.5 in the Systems/390 environment through the use of extended file information. In addition, new attributes will be built upon the Z390 attribute set in future releases.

–ATTRIB_COMPATIBILITY(Z390|MV25)

Although ZIP Archives created by older releases of ***PKZIP for MVS™*** can be processed by ***PKZIP for MVS™*** Version 5.5, extended attributes created by ***PKZIP for MVS™*** Version 5.5 in **Z390** mode will not be compatible with executions of older versions of ***PKZIP for MVS™***, and, they will be ignored, although the file data will still be accessible. For installations where multiple releases of the product will be run with files being shared between systems, a mode of **MV25** can be used so that the attributes created by release 5.5 are acceptable to the older product Version. However, new features and functions will only be generated in the **Z390** mode.

–CALLMODE

Synonyms Include: N/A

This command is an internal use command that is used for environmental interfacing and should not be specified.

–CALLMODE(BATCH|ISPF|TSO)

–CHECK_SYSIN_MEMBER

Synonyms Include: N/A

This is a defaults-module only parameter (since the value must be determined before the SYSIN command set is opened).

–CHECK_SYSIN_MEMEBER(Y|N)

The default operation of PKZIP MVS is to verify that command input stored in a PDS or PDSE member exists. If the member is not found, then a message is issued and the PKZIP function is terminated.

```
"ZPCM010E MEMBER NOT ACCESSIBLE IN DATASET"
```

Installations that use very large PDS/PDSE libraries may want to avoid the overhead of searching the directory. Performance may be improved by specifying –CHECK_SYSIN_MEMBER=N in the ACZDFLT module.

However, a system abend S013 will occur if the specified member does not exist in the library.

–COMPRESSION_LEVEL

Synonyms Include: –METHOD, –EN, –ES, –EX, –E0

This command specifies the speed and compression level when zipping a file.

–COMPRESSION_LEVEL(NORMAL|MAXIMUM|FAST|SUPERFAST|STORE)

When updating files in a ZIP archive, –COMPRESSION_LEVEL specifies a parameter that determines the compression level and speed to be used. The command may specify one of the five options.

The following table demonstrates the balancing effectiveness of compression versus speed of compression. The levels range from 0 (low) to 5 (high). For example, when using MAXIMUM, a resulting compressed file would be highly compressed, but it would take a longer time to complete.

Level	Compression	Speed
MAXIMUM	4	1
NORMAL	3	2
FAST	2	3
SUPERFAST (default)	1	4
STORE	0	5

Should the compressed file be larger than the original file, ***PKZIP for MVS™*** will perform a –COMPRESSION_LEVEL(STORE) on the file instead of compressing it. Note that this file processing overhead will require significantly more time than it would if a –COMPRESSION_LEVEL(STORE) command was issued initially.

–CRLF

Synonyms Include: –NOCRLF

☺ - Cross Platform Compatible command (VSE, MVS, OS/400, UNIX, and Windows).

☠ - Be aware that if this command is used incorrectly, you could incur problems. This command determines whether special delimiters or terminators are inserted when a file is being extracted from a ZIP archive.

–CRLF(Y|N|C)

Y - YES - Insert CR (carriage control), LF (line feed), or CZ (Ctrl-Z), as appropriate.

N - NO - Do not insert CR, LF, or CZ.

C - COMPATIBILITY - Changes the way PKZIP MVS processes the last record in a file.

When extracting a text file from a ZIP file that contains no internal delimiters or terminators of CR, LF, or CZ, you may use –CRLF(N) so that the PKUNZIP program creates fixed record lengths for the output. The maximum record length of the extracted dataset determines the output record length. The last record of the output is filled with EBCDIC spaces (Hex 40) if needed.

–FILE_TERMINATOR() and –DATA_DELIMITER() may be also be used and the PKUNZIP program will search for default delimiters.

See also –DATA_TYPE(TEXT).

When used in PKZIP program processing:

–CRLF=Y normally places the –DATA_DELIMITER character(s) after every record (including the last one) before conditionally adding the –FILE_TERMINATOR character(s).

–CRLF=C specifies that the last record should not have the –DATA_DELIMITER characters added after the last record of the file, and should only have the –FILE_TERMINATOR character(s) added.

Note: –CRLF(Y,NOEOFDELIM) also performs this action.

If the default values for –DATA_DELIMITER and –FILE_TERMINATOR are taken, the same output results will be seen with either CRLY=Y (standard) or CRLF=C. The advantage of using –CRLF=C or –CRLF(Y,NOEOFDELIM) is that finer control of the last control characters in the file can be achieved through the –FILE_TERMINATOR specifications.

When used in PKZUNZIP program processing:

–CRLF=C during an EXTRACT causes additional line control interpretation to be done when the –DATA_DELIMITER and –FILE_TERMINATOR characters specified do not accurately match the source file.

Use of –CRLF=C may cause records to be split when binary data (within a text file) is found to contain any of the typical line control characters.

ACZDFLT (MCZDFLTS macro):

When –CRLF=C is used in the MCZDFLTS macro and –FILE_TERMINATOR is not specified, the default for –FILE_TERMINATOR will be set to CRLFCZ(x'0D0A1A) instead of the standard default of CZ(x'1A'). This will yield equivalent ZIP results when –CRLF=Y is specified with its defaults.

"–FILE_TERMINATOR=" may be specified along with –CRLF=C to ZIP a file, resulting in no control characters at the end of the file.

If both CRLF=C and –FILE_TERMINATOR=CZ are specified, then –FILE_TERMINATOR=0D0A1A will be substituted. –FILE_TERMINATOR=1A may be used to override this substitution.

Processing examples:

	–DATA_DELIMITER CRLF = x'0A0D'	**–FILE_TERMINATOR CZ = x'CZ'**	
CRLF(N)	No control characters are inserted after any records.	No control characters are inserted at the end of the file.	Rec1_dataRec2_data…
CRLF(Y)	All records are terminated with –DATA_DELIMITER characters.	After the final record, the –FILE_TERMINATOR character is added.	Assuming the distribute defaults of: –DATA_DELIMITER=crlf –FILE_TERMINATOR=cz Rec1_data**CRLF** Rec2_data**CRLF** Last_record**CRLF** **CZ**
CRLF(C)	All records *except the last record* are terminated with –DATA_DELIMITER characters.	After the final record, the –FILE_TERMINATOR character is added.	Assuming the distribute defaults of: –DATA_DELIMITER=crlf –FILE_TERMINATOR=cz Rec1_data**CRLF** Rec2_data**CRLF** Last_record **CZ**
CRLF=Y,NOEOFDELIM	All records *except the last record* are terminated with	After the final record, the –FILE_TERMINATOR character is added.	Same as CRLF(C).

	–DATA_DELIMITER CRLF = x'0A0D'	–FILE_TERMINATOR CZ = x'CZ'	
	–DATA_DELIMITER characters.		

–DATA_DELIMITER

Synonyms Include: –DELIM

☺ - Cross Platform Compatible command (VSE, MVS, OS/400, UNIX, and Windows).

☠ - Be aware that if this command is used incorrectly, you could incur problems.

When used in PKZIP program processing:

When compressing a file as text (not binary), the –DATA_DELIMITER command specifies what character(s) to store at the end of each record to differentiate records. The last record does not get this delimiter but instead gets the characters specified by the –FILE_TERMINATOR command. When compressing a file as binary, the –DATA_DELIMITER command is ignored.

–DATA_DELIMITER(<delim chars>)

Delim chars are the delimiter characters to be appended. There may be 0-4 characters specified in any combination:

CR - Appends an ASCII Carriage Return (hex 0D).

CZ - Appends an ASCII Ctrl-Z character (hex 1A).

LF - Appends an ASCII Line Feed character (hex 0A).

() - No delimiters at all.

The default is CRLF if no –DATA_DELIMITER command is specified.

Note: Transfers of Microsoft- Disk Operating System (MS-DOS) records use a CRLF for a delimiter, while UNIX records use a LF.

When extracting the file(s), the same –DATA_DELIMITER command should be used differentiate each record, just as it was when it was compressed.

PKZIP for MVS™ searches for one each of CR, CZ, and LF characters as a default for text file record delimiters. If a file was compressed with double characters as delimiters, for example, –DATA_DELIMITER(**LF**CZ**LF**), and the file was later decompressed without the –DATA_DELIMITER command (a default search is used), ***PKZIP for MVS™*** would use each LF as a record delimiter. It would then create extra record(s) to accommodate for the duplicate characters, for example, **LF**.

When used in PKUNZIP program processing:

When decompressing a text file (not binary), the –DATA_DELIMITER command specifies what characters to look for at the end of records (except the last) that serve as delimiters. The delimiter is removed from the record when it is decompressed. The last record of the file ends with the characters specified in the –FILE_TERMINATOR command. When decompressing a binary file, the –DATA_DELIMITER command is ignored.

–DATA_DELIMITER(<delim chars>)

delim chars - The delimiter characters to be appended. There may be 0-4 characters specified in any combination:

CR - Appends an ASCII Carriage Return (hex 0D).

CZ - Appends an ASCII Ctrl-Z character (hex 1A).

LF - Appends an ASCII Line Feed character (hex 0A).

() - No delimiters at all.

The default is CRLF if no –DATA_DELIMITER command is specified.

> **Note: Transfers of Microsoft- Disk Operating System (MS-DOS) records use a CRLF for a delimiter, while UNIX records use a LF.**

Default processing of records. ***PKZIP for MVS™*** will search for a range of delimiters when the –DATA_DELIMITER command is not used. They are: CRLFCZ, LFCRCZ, CRLF, LFCR, CRCZ, LFCZ, CR, and LF. This default may be used unless special delimiter combinations were assigned during compression. To assure correct location of records, the same –DATA_DELIMITER command used in compression should be used to decompress as well.

–DATA_STORAGE

Synonyms Include: –CACHEMEMORY

Cache memory may be specified, with the –DATA_STORAGE command, in order to increase processing speed. This command specifies the total number of bytes to be allocated for caching. The default is zero (0) - no caching is performed - when this command is not specified.

```
-DATA_STORAGE(<bytes>)
```

bytes - This specifies the total number of bytes assigned for caches in ***PKZIP for MVS™***. Where <bytes> may range from 64000 to (2^{31} - 1).

The unit is specified only in bytes and with no commas, therefore to specify a cache memory of 0.30Mb, use –DATA_STORAGE(300000).

A larger file may be processed in less time by specifying a larger cache memory. A larger cache memory would increase virtual memory for compression operations that may decrease the necessary number of disk accesses. This would reduce I/O time and therefore improve compression performance time.

Warning: Be aware of your system's storage requirements before specifying very large amounts of cache memory, for example, 1 Gbyte or 1000000000. Claiming too large a cache memory when there is insufficient storage and/or Page Datasets can create serious problems. It is suggested that one should verify this capability with the proper systems management personnel before attempting a command of this magnitude.

PKZIP for MVS™ can use multiple caches during processing which can vary the actual amount of virtual memory that is used. The amount is affected by the number and size of the files being processed.

–DATA_TYPE

Synonyms Include: –DETECT, –BINARY, –TEXT, –DETECTX

This command specifies that files for compression are either binary, text, or detectable. If the modifier is (BINARY), then no translation will be performed on the files. If the modifier is (TEXT), then files selected for compression are text files and will be translated from EBCDIC to ASCII before compression. If neither of these are specified, the program will make a determination (DETECT) based on the existing data type. The program will read in a portion of the data, evaluate it, and determine the appropriate process.

```
–DATA_TYPE(DETECT|BINARY|TEXT|DETECTX)
```

If you know the file type, you will save processing time by specifying –DATA_TYPE(BINARY), –DATA_TYPE(TEXT), or –DATA_TYPE(BINARY) with –SAVE_LRECL(Y).

When used in PKZIP program processing:

and when specifying –DATA_TYPE(BINARY):

No translation of the data will be performed and record terminators will not be inserted. A binary file contains no delimiters between records and should only be used when the target system (for UNZIP) will be able to handle the EBCDIC format. Variable length files should be processed with the addition of the –SAVE_LRECL(Y) command. This is commonly used when exchanging files between Systems/390 operating environments, for example, Load Modules.

and when specifying –DATA_TYPE(TEXT):

A compressed text file is stored as ASCII, (unless otherwise specified with –TRANSLATE_TABLE_DATA), and stored with the specified delimiters (–DATA_DELIMITER) and terminator (–FILE_TERMINATOR). Note that the translation defaults and delimiter and terminator defaults of a stored text file from ***PKZIP for MVS™*** make the file compatible with compressed files on other platforms. As a result, compressed text files may be extracted, onto other platforms.

and when specifying –DATA_TYPE(DETECT) or –DATA_TYPE(DETECTX) :

PKZIP MVS™ will attempt to dynamically determine whether the data should be translated into TEXT format. A portion of the file (see –DATATYPE_DETECT_DEPTH) is examined using the tailorable DETECTXT translation table (see –DATATYPE_DETECT_TABLE) and compared to the value specified in –DATATYPE_TEXT_PERCENT.

When used in PKUNZIP program processing:

when specifying –DATA_TYPE(BINARY):

If the raw format of the data is desired, regardless of whether the originating system ZIPPED the file as TEXT, use this command.

Binary processing will not attempt to resolve record delimiters. As a result, the data will be streamed into records according to the file allocation specifications. Note that when using ***PKZIP MVS™*** to create binary files that are targetted for another ***PKZIP MVS™*** system, –SAVE_LRECL(Y) can be specified to preserve record lengths.

when specifying –DATA_TYPE(TEXT):

The selected file will be treated as a text file regardless of the Archive directory indicator for the file. This can be used when the originating system is known to have ZIPPED an ASCII text file as binary. To discover what file type exists in the Archive directory entry, see the –ACTION(VIEW) command.

When the PKUNZIP program extracts the selected file, it first translates the character set and then it extracts records to the output file as determined by imbedded record delimiters. (See –DATA_DELIMITER command). The delimiters are not included in the extracted file. If the output file is a fixed record length, then records that exceed the record length will be truncated and records that are smaller than the record length will be filled with EBCDIC spaces (hex 40).

If no delimiters are imbedded in the selected file, the command –CRLF(N) should also be used. This command directs the PKUNZIP program to not seek out record delimiters but instead use the maximum record size in creating the output.

when specifying –DATA_TYPE(DETECT):

The PKZIP Archive layout contains an indicator that reflects whether the file was ZIPPED as text. ***PKZIP MVS™*** honors that flag when DETECT is specified. This is the default setting. However, there are cases

that DETECTX is recommended when TEXT data has been ZIPPED in an ASCII environment with a binary indication, for example, a workstation ZIP compatible product is used to create the Archive.

when specifying –DATA_TYPE(DETECTX) :

On some platforms, for example, workstations, some ZIP utilities do not set the TEXT indicator although the data was ASCII text. In this situation, DETECTX is recommended so that ***PKZIP MVS™*** will attempt to dynamically determine whether the data should be translated into EBCDIC TEXT format. A portion of the file (see –DATATYPE_DETECT_DEPTH) is examined using the tailorable DETECTXT translation table (see –DATATYPE_DETECT_TABLE) and compared to the value specified in –DATATYPE_TEXT_PERCENT. (Note that the detection depth will be limited in size to the first internal buffer being extracted. This will typically be less than 64K).

–DATATYPE_DETECT_DEPTH

Synonyms Include: –DATATYPE_SCAN_DEPTH, –DETECT_DEPTH

This command specifies the distance that a file is scanned before making a determination as to whether it is binary or text. It can be specified as a number of records (1000R) or as a size in bytes (64000), Kilobytes (64K), or Megabytes (4M).

–DATATYPE_DETECT_DEPTH(<amt>)

amt:

amount in records (1000R).
amount in bytes (64000).
amount in kilobytes (64K) (8K is the default).
amount in megabytes (4M).

It is important to note that the amount of data specified in this parameter will be buffered in virtual storage during the text/binary translation period, and before the data is directed to the compression algorithms. (Compression cannot be performed until data translation and record delimiter processing is done, which follows DATA_TYPE detection). The buffering is done for performance reasons to avoid Close/Open/ Re-read overhead). However, sufficient virtual storage (31-bit region) must be available to temporarily hold the specified quantity, or storage capacity issues may arise.

Note that in UNZIP processing with DETECTX, Record count is changed to a maximum setting of 64K. Since the data is being scanned for ASCII characters before record processing is determined, a record count is not applicable. The amount of data scanned for DETECTX is also limited to the amount of data returned by the decompression engine (typically a maximum of 64K), and will be dynamically rounded down as needed.

–DATATYPE_DETECT_TABLE

Synonyms Include: N/A

This command specifies the table of characters used to assess whether a byte is text or binary. The default table name is DETECTXT.

–DATATYPE_DETECT_TABLE(<tablename>|DETECTXT)

tablename - This is a tablename of characters used to assess whether a byte is text or binary.

DETECTXT - This is the default table as shipped with the product.

The specified TRANSLATE and TEST table is used to detect binary data within data records when –DATA_TYPE(DETECT) is specified for ZIP processing.

The table is used as a character lookup table for each byte scanned through –DATATYPE_SCAN_DEPTH. The binary value of each data byte is used to locate a position in the table. If the table position is x'00', then that byte is considered to be BINARY. If the table position is NONZERO, then the byte is counted as TEXT. The actual value in the table is not important, but the locations have been filled in with the equivalent offset for ease of editing (the comments reflect the character value where possible, although some bytes (such as CR/LF) are simply indicated with a comment of ".").

This table may be changed, copied, and re-assembled to adjust for data dependencies. The table used (loaded as a load module) is specified in –DATATYPE_DETECT_TABLE and may be specified either in the defaults module or by command (Seemembers in `INSTLIB(ASMDETXT) and (DETECTXT)`.

–DATATYPE_TEXT_PERCENT

Synonyms Include: N/A

This command specifies the percentage of the sample that must meet the “text” criteria before it will be considered to be TEXT.

–DATATYPE_TEXT_PERCENT(<percent>)

percent - This is the percentage from 1-100 is required (97% is the default).

If the entire file is read before –DATATYPE_DETECT_DEPTH is reached, then the percentage will be computed according to the number of bytes read. For example, if –DATATYPE_TEXT_PERCENT=97 was specified, with –DATATYPE_DETECT_DEPTH=64K, then .03 * (64*1024) = 1966 (rounded down). Once 1967 binary characters are found, then the entire DEPTH cannot meet 97% text, so the scan is terminated and the file is marked as BINARY.

Given the percentage above (97%), if a file has 100 records, each containing 80 bytes of text with 2 bytes of additional termination information (total 82 bytes), then this file would pass as TEXT. 100 * 82 (8200) * .03 = 246 Thus, 246 bytes of binary data would be required to mark this file as BINARY, but there are only 200.

–DDNAME_PARMLIB

Synonyms Include: N/A

This command specifies the name of the JCL DD statement used to read the preset commands which are read before the //SYSIN member.

–DDNAME_PARMLIB(<ddname>)

ddname - This is the DDname of the preset parameters member.

PARMLIB - This is the default DDname.

–DDNAME_SYSIN

Synonyms Include: N/A

This command specifies the name of the JCL DD statement used to identify the SYSIN member. It can go into the defaults module to specify which DDname to open to read job level commands.

–DDNAME_SYSIN(<ddname>)

ddname - This is the DDname of the SYSIN member.

SYSIN - This is the default DDname.

–DDNAME_SYSPRINT

Synonyms Include: N/A

This command specifies the name of the JCL DD statement used to identify where messages will be written.

–DDNAME_SYSPRINT(<ddname>)

ddname - This is the DDname of the SYSPRINT member.

SYSPRINT - This is the default DDname.

–DDNAME_ZPSORTIN

Synonyms Include: N/A

This command specifies the name of the JCL DD statement used for sorting directory information associated with –VIEW processing. This should not need to be changed unless the name conflicts with other JCL allocation used in the same job step.

–DDNAME_ZPSORTIN(<ddname>)

ddname - This is the DDname to use for SORTIN.

ZPSRTIN - This is the default DDname.

Note: The value specified for –TEMP_UNIT is used to allocate a temporary work file with this DD.

–DDNAME_ZPSORTOUT

Synonyms Include:

This command specifies the name of the JCL DD statement used for sorting directory information associated with –ACTION(VIEW) processing. This should not need to be changed unless the name conflicts with other JCL allocation used in the same job step.

–DDNAME_ZPSORTOUT(<ddname>)

ddname - This is the DDname to use for SORTOUT.

ZPSRTOUT - This is the default DDname.

–ECHO

Synonyms Include: –NOECHO

Commands used for the PKZIP and PKUNZIP programs are put into the output message dataset when –ECHO(Y) is specified. This is the default setting.

–ECHO(Y|N)

Y - YES - Log all output messages to SYSOUT.

N - NO - Do not log output messages to SYSOUT.

One would use –ECHO(Y) if the –ECHO(N) command had previously been used (either in the configuration module or through the JCL) to suppress output messages. Then the commands that are output begin with the –ECHO(Y) command itself. Since the –ECHO command is processed before it is activated, errors in this line would not appear in the output message dataset.

–ENCRYPTION_METHOD

Synonyms Include: -AES128 | -AES192 | -AES256

☺ - Cross Platform Compatible command (MVS, OS/400, UNIX, and Windows).

☠ - Be aware that if this command is used incorrectly, you could incur problems.

When a ZIP action is requested to save a file in an Archive and a password is provided, ***PKZIP for MVS™*** will use an encryption method to protect the data. This command value specifies which algorithm is to be employed.

Standard - This algorithm is the original algorithm used in PKZIP 2.x products and is compatible with other PKZIP 2.04g products that support standard encryption. This is the default value for the product unless the installation defaults module has been tailored differently.

AES128 - A PKZIP exclusive implementation of the AES 128-bit key algorithm (also known as Rijndael) will be used.

AES192 - A PKZIP exclusive implementation of the AES 192-bit key algorithm (also known as Rijndael) will be used.

AES256 - A PKZIP exclusive implementation of the AES 256-bit key algorithm (also known as Rijndael) will be used.

Usage Notes:

- This algorithm is part of the Strong Encryption Suite provided by PKWARE, Inc.
- PKUNZIP will automatically detect which encryption method was specified during the ZIP process and operate accordingly.
- During a PKZIP (ZIP) run, only 1 encryption method may be specified, and that method will be used for each file operated on.
- By executing PKZIP at different times, various files within the Archive may be saved with differing levels (and types) of protection. That is, some files may not be protected at all, while others may have different methods and/or passwords.
- A “+” is shown in a View to indicate Standard Encryption protection is used for a file.
- A “!” is shown in a View to indicate Strong Encryption protection is used for a file.
- When specifying long passwords (requiring multiple control records) do not use the “+” continuation character (because it supplies an implicit blank in the command stream).
- This enhanced feature for ADD, UPDATE, and FRESHEN applies to standard PKZIP Archives and not GZIP.

–EXCLUDE(dsname mask)

Synonyms Include: N/A

This parameter has no equivalent. It is a new command.

When selecting a large number of files via a mask selection it may be useful to eliminate some of the files from being processed, for example, GDGs, ZIP archives, or other special files that can be identified by their dataset naming conventions.

See also: –SELECT_TAPE, –SELECT_VSAM, –SELECT_CATALOGED_ALIAS, and –RECALL_TO_ZIP for other selection-restricting capabilities.

The dsname mask may be a fully qualified file name or a masked name (similar to dataset selection names) of 1 to 80 characters. (Embedded blanks in an MVS dsname for ZIP processing will truncate the mask.)

Multiple –EXCLUDE commands may be specified in an execution. A table is built from all of the commands found and is scanned for a match against a candidate file for selection. The file will be excluded if ANY of the masks is a match.

Note that there is no default for this command, nor can one be specified in the ACZDFLT module. This is a run-time only command, although it may be specified through the PARMLIB DD or EXEC parms (including a parm string from a calling program) in addition to SYSIN.

Example:

Assume that PDS SYS1.PARMLIB contains members CLOCK01, CLOCK02, CLOCK11, and CLOCK13. If the following commands were issued for PKZIP:

SYS1.PARMLIB(CLOCK*)

–EXCLUDE=SYS1.PARMLIB(*11)

Member CLOCK11 would be excluded from the ZIP process, while the other members would be processed.

–EXTRACT_PREVIEW

Synonyms Include: –PREVIEW

When the contents of a large archived file is unknown, it may be useful to extract a small portion of the file for the purpose of previewing the data. The –EXTRACT_PREVIEW(nnnnnnnn) command limits the number of records to extract and can save a considerable amout of time in assessing data content.

–EXTRACT_PREVIEW(<nnnnnnnn>)

The parameter value specifies the maximum number of records to extract. If the value is either 0 (or not supplied) then the entire file is extracted.

–FILE_BUSY_WAITTIME

Synonyms Include: N/A

This command specifies how long ***PKZIP for MVS™*** should wait while continually retrying before it will terminate and give an error message or go on to further processing.

–FILE_BUSY_WAITTIME(<HHMMSSTH>)

HHMMSSTH:

HH - Hours
MM - Minutes
SS - Seconds
T - Tenths of a second
H - Hundredths of a second

00100000:

10 minutes is the default

–FILE_EXTENSION

Synonyms Include: –CNVEXT

☺ - Cross Platform Compatible command (VSE, MVS, OS/400, UNIX, and Windows).

☠ - Be aware that if this command is used incorrectly, you could incur problems.

When a file is extracted and the archive name contains an extension, the –FILE_EXTENSION command specifies what to do with the extension. There are three options: DROP (the default), SUFFIX, or NAMEFILE.

–FILE_EXTENSION(DROP|SUFFIX|NAMEFILE)

DROP - This is the extension (which will drop the last data level of the Archive File).

Example:	
Given the file:	FIRST/RATE/DATES/TEST
and a command of:	-FILE_EXTENSION(DROP)
the file will be:	FIRST.RATE.DATES
the PDS will be:	FIRST.RATE(DATES)

SUFFIX - This is the extension to the last data level (note that any generated name longer than 8 characters will be truncated to 8 characters).

Example:	
Given the file:	FIRST/RATE/DATES/TEST
and a command of:	-FILE_EXTENSION(SUFFIX)
the file will be:	FIRST.RATE.DATESTES
the PDS will be:	FIRST.RATE(DATESTES)

NAMEFILE - This is the extension into a data level.

Example:	
Given the file:	FIRST/RATE/DATES/TEST
and a command of:	-FILE_EXTENSION(NAMEFILE)
the file will be:	FIRST.RATE.DATES.TEST
the PDS will be:	FIRST.RATE.DATES(TEST)

The default is **DROP** if no –FILE_EXTENSION command is specified.

–FILE_TERMINATOR

Synonyms Include: –TERM

☺ - Cross Platform Compatible command (VSE, MVS, OS/400, UNIX, and Windows).

☠ - Be aware that if this command is used incorrectly, you could incur problems.

–FILE_TERMINATOR(<delim chars>)

delim chars - These are the delimiter characters to be appended. There may be 0-4 characters specified in any combination:

CR - Appends an ASCII Carriage Return (hex 0D).

CZ - Appends a ASCII Ctrl-Z character (hex 1A).

LF - Appends a ASCII Line Feed character (hex 0A).

() - No delimiters at all.

When used in PKZIP program processing:

When compressing a file as text (not binary), the –FILE_TERMINATOR command specifies what character(s) to store at the end of the last record of the file to signal the end. When compressing a file as binary, the –FILE_TERMINATOR command is ignored.

When used in PKUNZIP program processing:

When decompressing a text file (not binary), the –FILE_TERMINATOR command specifies what character(s) to find at the end of the last record of the file to signal the end. When decompressing a binary file the –FILE_TERMINATOR command is ignored.

When used in either type of processing:

The default is CRLFCZ if no –FILE_TERMINATOR command is specified with the PKZIP program. With the PKUNZIP program the default is CRLFCZ if no characters are specified by –FILE_TERMINATOR(). Otherwise a range of standard delimiters are used in the search which should satisfy most systems.

MS-DOS records use CRLFCZ for a delimiter.

UNIX records use LF for a delimiter.

When extracting the file(s), the same –FILE_TERMINATOR command that was used to ZIP should be used to UNZIP to process the file correctly if non-standard delimiter characters were used.

The –FILE_TERMINATOR characters should be different than the –DATA_DELIMITER characters to make the last record distinct. Use a different combination of characters to create the distinction.

For example, this would not be distinct:

–DATA_DELIMITER(CRLF).
–FILE_TERMINATOR(**CR**LFCZ) Note the same CRLF in the character set.
or
–FILE_TERMINATOR(CZ**CRLF**) where a single record of CZ would be created.

This would be distinct:
–DATA_DELIMITER(CRLF).
–FILE_TERMINATOR(CZCRCZ) there is no duplication of character sets.

–GDGALL_SUPPORT

Synonyms Include: –GDGALL, –NOGDGALL, –SELECT_GDGALL

This command determines whether all levels of a Generation Data Group (GDG) are retrieved and included in the archive.

```
–GDGALL_SUPPORT(Y|N)
```

Y - YES - All levels of the dataset are retrieved.

N - NO - Only the current dataset (Level 0) is retrieved.

–GZIP

Synonyms Include: –NOGZIP

☺ - Cross Platform Compatible command (MVS, VSE, OS/400, UNIX, and Windows).

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command may be used to create an archive using the GZIP-compatible format instead of the ZIP format. For further information, see Chapter 10 - Processing with GZIP.

–GZIP(Y|N)

Y - YES - Process using GZIP-compatible ***PKZIP for MVS™*** processing.

N - NO - Process using normal ***PKZIP for MVS™*** processing.

–GZIP_SUFFIX

Synonyms Include: N/A

☺ - Cross Platform Compatible command (MVS, VSE, OS/400, UNIX, and Windows).

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command may be used when there is no valid GZIP filename. The archive input file name will be used and the last level of the name will be replaced with the value of this field.

–GZIP_SUFFIX(<*suffix*>)

suffix - The name to be used as the last level of the filename.

–HIERARCHY

Synonyms Include: –NOHIERARCHY

–HIERARCHY(Y|N)

Y - YES - Specifies that the entire dataset name stored in the ZIP archive file is to be used to convert the file to an MVS format.

N - NO - Strips away higher level components and uses the lowest level of the dataset component(s) as the member name when creating a file name in the PDS. It is used when converting a file from ZIP archive format to MVS format. The PDS should be specified with the command –SELECT_FROM_PDS or ZIPCUR.

Example:	
Given the file:	TDS/DICT/DATA
and a command of:	–HIERARCHY(Y)
the file will be:	TDS.DICT.DATA
the PDS will be:	TDS.DICT(DATA)

Example:	
Given the file:	TDS/DICT/DATA
and a command of:	-ZIPCUR (MYRE.SPELL.CHK) -HIERARCHY(N)
the PDS will be:	MYRE.SPELL.CHK(DATA)

If the PDS member already exists, you must replace it with –OUTFILE_OVERWRITE or add it with –INSERT_MEMBER to keep the member.

–INFILE

Synonyms Include: –INDD, –IFILE,–INFILE_DD

The –INFILE command identifies the DD statement that further describes the file to be compressed.

–INFILE(<*ddname*>[,member1][,member2][,…member*n*])

ddname - This is the name of the DD job step listed in the JCL.

Member1-n - This is 0 to n member names that identify specific members within the PDS (described in the <ddname> used in the job step).

The DD statement may describe a sequential dataset, an entire PDS or a member of a PDS, or even a generation of a GDG.

If a member of a PDS is to be compressed, there are two methods of identifying that member.

First, using just the DD statement where the individual member is described in the DD statement and –INFILE refers to that DD statement.

```
Example:  //INPUT      DD  DISP=SHR, DSN=MY.DATA.FILES(.FIRST)
    .
    .
    .
    -INFILE(INPUT)
```

Second, using the command where the entire PDS is described in the DD statement and then the –INFILE command refers to that DD statement as well as the individual member name(s) to use.

```
Example   //INPUT      DD  DISP=SHR, DSN=MY.DATA.FILES

    .
    .
    -INFILE(INPUT,FIRST, SECOND,FIFTH)
```

Note that more than one member may be indicated with the command.

If no members are indicated, the entire PDS is used.

Multiple –INFILE commands can be used.

See <dataset name> for dataset naming capabilities.

–INSERT_MEMBER

Synonyms Include: –INSERTMEMBER, –NOINSERTMEMBER

The –INSERT_MEMBER command is used to add a member to an existing PDS.

–INSERT_MEMBER(Y|N)

Y - YES - Specifies that the newly extracted member will be added and become a new member of an existing dataset.

N - NO - Specifies that the member will not be added and the process will fail with an error message.

See –OUTFILE_OVERWRITE to update a dataset in an existing PDS.

–KEY_PROTECT_LEVEL

Synonyms Include: –KEYPROTECT1, –KEYPROTECT2

This parameter has no equivalent. It is a new command.

–KEY_PROTECT_LEVEL(1|2)

When using advanced encryption (see –ENCRYPTION_METHOD) during a ZIP operation, additional information is stored in the ZIP Archive pertaining to the encryption keys. This information is also encrypted to further secure the file data.

The use of this parameter will affect the size of the resulting Archive. –KEY_PROTECT_LEVEL(1) will use approximately 100 more bytes per file in the Archive, while –KEY_PROTECT_LEVEL(2) will require 340 more bytes per file.

–LICENSE_HLQ

Synonyms Include: N/A

This command specifies the high level qualifier to be used in locating the License Control Dataset. This should be specified in accordance with directions provided by the Systems Programmer responsible for setting up the product and maintaining its licensing options. It will be used to allocate the pkzipmvs.LICENSE dataset during execution.

See Appendix A - Licensing Requirements for more information.

–LICENSE_HLQ(<*hlvl*>)

hlvl - High level qualifier used for allocation (PKZIP.MVS is the default).

–LMOD_SUPPORT

Synonyms Include: N/A

☺ - This is an MVS command only.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command determines whether ***PKZIP for MVS™*** processing will dynamically turn on the commands of –DATA_TYPE(BINARY), –SAVE_LRECL, and –SAVE_FILE_ATTRIBUTES for PDS members that have been detected as Load Modules. These modules will then be reconstructed during UnZip processing.

This feature allows text-based non-load module files to be zipped during a single pass along with load libraries. One might use this feature to process a PDS containing both load modules and text files in a single pass.

–LMOD_SUPPORT(Y|N)

Y - YES - Turn on –LMOD_SUPPORT support. Zip processing will dynamically turn on –SAVE_LRECL and –DATA_TYPE(BINARY) for PDS members detected as being load modules.

N - NO - Do not turn on –LMOD_SUPPORT support.

See –DATA_TYPE(BINARY), –SAVE_FILE_ATTRIBUTES, and –SAVE_LRECL for additional information.

–LOGGING_LEVEL

Synonyms Include: –VERBOSE, –Q, –QUIET

This command specifies the level (or quantity) of messages that will be output from ***PKZIP for MVS™*** to SYSPRINT.

–LOGGING_LEVEL(NORMAL|QUIET|VERBOSE)

NORMAL - Specifies that a standard set of messages will be output to SYSPRINT.

QUIET - Specifies that no messages are issued, although return codes will be set when errors occur. This option is normally used when calling from a CLIST or another program where you will not immediately view the output (as in ISPF execution).

VERBOSE - Specifies that a more detailed level of messages will be output to provide in-depth processing information.

Dynamic Allocation Parameters used to create and access files.

Dynamic Allocation error codes.

System SORT messages (–TRACE_SORT (4) may provide more).

Specific PKZIP messages.

–MEMORY_MODEL

Synonyms Include: –MEM_MODEL, –MEM_MDL, –SMM, –MMM, –LMM, –MMS, –MML

–MEMORY_MODEL(SMALL|MEDIUM|LARGE)

–MEMORY_MODEL(SMALL|MEDIUM|LARGE) controls where file management control blocks are held, such as, control blocks describing an archive file with its attributes, and the amount of storage than can be used for compression control tables.

When –MEMORY_MODEL(LARGE) is specified or defaulted, all of the file management control blocks are held in 31-bit virtual storage and the largest compression tables are used (providing the best compression possible for the Compression_Level selected).

When either SMALL or MEDIUM are specified, the file descriptor information is spilled to a set of work files to be sorted, merged and selected. Note that file descriptors are built for both files existing in the input archive and new files to be selected, so the aggregate count must be managed. Approximate sizes for each file descriptor are as follows:

VSAM - 2.5K.

Sequential - 800 bytes.

PDS/PDSE - 800 bytes for base dataset + 224 bytes per member.

–MULTI_THREAD_LIMIT

Synonyms Include: –TASKS

To specify more than one task to be used while compressing a dataset(s), use –ARCHIVE_MGMTCLASS. Some systems have more that one CPU and can run subtasks to aid in processing. The compression of a dataset would then run with two or more subtasks (depending upon the specified amount). These subtasks would run in parallel and speed processing time, improving performance for the processing of multiple datasets.

–MULTI_THREAD_LIMIT(<amount>)

amount - Specifies the maximum number of subtasks that may be used by ***PKZIP MVS™*** to compress datasets. The <amount> should not exceed twice the number of CPU's on a system. Should this command be used on a single CPU system, the results are undefined. An amount of 3 is the default.

ZIP processing speed can improve with this command, however, actual performance is dependent on the type of datasets that are processed.

Datasets within a PDS are processed within the same subtask unless the datasets are individually identified in separate dataset definitions.

–NOSYSIN

Synonyms Include: –NOSYSIPT

–NOSYSIN

The SYSIN dataset will not be opened if –NOSYSIN is specified in the PARM parameter or in the Configuration file. The command has no effect if placed within the SYSIN dataset.

This command is useful when calling the PKZIP program from another program. The SYSIN passed to the calling program will not be effected by the PKZIP program processing in this situation. See Invoking ***PKZIP for MVS™*** Services.

–ON_FILE_ACCESS_ERROR

Synonyms Include: –FILESELERR

When used in PKZIP program processing:

If an access problem occurs during the ZIP processing of an input file or a temporary archive, the –ON_FILE_ACCESS_ERROR command specifies whether to terminate processing or ignore the error and continue. The default is to allow for compatibility.

–ON_FILE_ACCESS_ERROR(STOP|TERMINATE|TOLERATE|
IGNORE[,WARNONBUSY]|[,WARNIFBUSY])

STOP - Processing halts when an access error is detected.

TERMINATE - Processing halts when an access error is detected.

TOLERATE - Processing continues with the next file. Error return codes and messages of the problem files are produced.

IGNORE - Processing continues with the next file. Error return codes and messages of the problem files are produced.

WARNIFBUSY - Processing continues with the next file, and the "busy" files are reported as a warning. This is an option to STOP or TOLERATE. Without STOP or TOLERATE specified, busy files will be skipped.

WARNONBUSY - Same as WARNIFBUSY.

> **Note: This is different from a similar command, –ON_FILE_IO_ERROR which refers to file errors during a read.**

–ON_FILE_IO_ERROR

Synonyms Include: –FILEPROCERR

When used in PKZIP program processing:

If an I/O problem occurs during ZIP processing of an input file or a temporary archive, the –ON_FILE_IO_ERROR command specifies whether to terminate processing or ignore the error and continue.

–ON_FILE_IO_ERROR(STOP|TERMINATE|TOLERATE|IGNORE)

STOP - Processing halts when an I/O error is detected.

TERMINATE - Processing halts when an I/O error is detected.

TOLERATE - Processing continues with other files. Should all files receive errors, the archive may be empty as no file processing occurred.

IGNORE - Processing continues with other files. Should all files receive errors, the archive may be empty as no file processing occurred.

In either case, the PKZIP program will create a return code and error message(s) indicating the problem.

> **Note: This is different from a similar command, –ON_FILE_ACCESS_ERROR which refers to file errors during access, before the file is read.**

When used in PKUNZIP program processing:

If an I/O problem occurs during ZIP processing of an output file, the –ON_FILE_IO_ERROR command specifies whether to terminate processing or ignore the error and continue.

–ON_FILE_IO_ERROR(STOP|TERMINATE|TOLERATE|IGNORE)

STOP - Processing halts when an I/O error is detected.

TERMINATE - Processing halts when an I/O error is detected.

TOLERATE - Processing continues with other files following an I/O error.

IGNORE - Processing continues with other files following an I/O error.

In either case, the PKUNZIP program will create a return code and error message(s) indicating the problem.

> **Note: This is different from a similar command, –ON_FILE_ACCESS_ERROR which refers to file errors during access, after the file is extracted.**

–OUTFILE_BLKSIZE

Synonyms Include: –OUTBLKSIZ, –OUTBLKSIZE

The –OUTFILE_BLKSIZE command specifies a block size to be used when extracting a new dataset (but not for a dataset with an unblocked format).

–OUTFILE_BLKSIZE(<block size>)

block size - The block size to be used for a newly extracted dataset.

If the block size is not specified by this command, the size is taken from the information stored in the archive. If neither is available, a default size of 6160 bytes is set (assuming that an –OUTFILE_DATACLASS was not specified, in which case the default is not used).

A value of zero will cause the PKUNZIP program to calculate a block size for sequential or PDS files. However, do not use a value of zero for undefined files (–OUTFILE_RECFM(U)) as the resulting calculated block size may not be appropriate.

–OUTFILE_DATACLASS

Synonyms Include: –OUTDCLASS

This command pertains to DF/SMS allocation of new files when doing PKUNZIP processing. If you specify these classes, they will be passed to DF/SMS when dataset allocation occurs.

–OUTFILE_DATACLASS(<SMS Data Class>)

See IBM's DF/SMS manuals for further information about this parameter.

–OUTFILE_DD

Synonyms Include: –OUTDD, –OFILE, –OUTFILE, –OUT_FILE

The –OUTFILE_DD command identifies the DD statement that further describes the dataset into which the files are to be extracted.

–OUTFILE_DD(<*ddname*>)

ddname - This is the DD statement in the JCL that identifies the dataset to which files are extracted. When using –OUTFILE_DD, allocation and attribute information should be provided in the JCL for the output file.

Multiple –OUTFILE_DD commands may not be used.

Other UNZIP commands are related to the function of –OUTFILE_DD and may not be needed when –OUTFILE_DD is used. They are the following:

–FILE_EXTENSION - Specifies DROP, SUFFIX, or NAMEFILE to tell what to do with file extensions when extracting. The DD statement will determine the name of the output file.

–OUTFILE_SPACE_RLSE - Specifies what to do with free space allocation. This will already be determined in the DD statement.

–OUTFILE_LRECL, –OUTFILE_BLKSIZE, and –OUTFILE_RECFM - Specifies record length, block size, and file type, respectively. This will already be determined in the DD statement.

–UNZIPPED_DSN - Specifies exactly what files are to receive the extracted data. This file is determined in the DD statement, but the member name may be affected with the –UNZIPPED_DSN command in operation.

–OUTFILE_DIR_BLOCKS

Synonyms Include: –OUTDIRBLKS, –OUTFILE_DIRBLKS

This command specifies the number of directory blocks to be used when a PKUNZIP process requires that a partitioned dataset (PDS) is to be created. When –OUTFILE_DSNTYPE is PDS or extended attributes are used to create the output file, then –OUTFILE_DIR_BLOCKS can be used to specify or override the number of directory blocks to be allocated.

–OUTFILE_DIR_BLOCKS(<blocks>)

blocks - This is an 8-character field specifying the number of directory blocks to be allocated for a partitioned dataset.

00000010 - Ten directory blocks is the default.

–OUTFILE_DSNTYPE

Synonyms Include: –OUTFILE_DSORG, –OUT_DSORG, –MAKEPDS, –MAKEPDSE, –MAKELIBRARY, –MAKESEQ, –MAKEVSAM, –MAKEESDS

The –OUTFILE_DSNTYPE command determines the type of output file to be created. This command overrides any stored file attributes.

–OUTFILE_DSNTYPE(SEQ|PDS|PO|PDSE|LIBRARY|VSAM)

If the modifier is SEQ:

The extracted file will be a sequential dataset.

Example:	
Given the ZIP file:	MY/DATA/SOURCE/ACCOUNTS
and a command of:	–OUTFILE_DSNTYPE(SEQ)
the extracted file will be:	MY.DATA.SOURCE.ACCOUNTS

If the modifier is PDS, PDSE, PO, or LIBRARY:

The extracted file will be a partitioned dataset. The member name comes from the lowest level of the source dataset name. If the PDS receiving the file already exists, you must specify –INSERT_MEMBER(Y) or –OUTFILE_OVERWRITE(Y) to determine what to do with the additional PDS file.

Example:	
Given the ZIP file:	MY/DATA/SOURCE/ACCOUNTS
and a command of:	–OUTFILE_DSNTYPE(PDS)
the extracted member will be:	MY.DATA.SOURCE(ACCOUNTS)

If the modifier is VSAM:

The extracted file will be a VSAM file.

Example:	
Given the ZIP file:	MY/DATA/SOURCE/ACCOUNTS
and a command of:	-OUTFILE_DSNTYPE(VSAM)
the extracted cluster name will be:	MY.DATA.SOURCE.ACCOUNTS

–OUTFILE_LRECL

Synonyms Include: –OUTLRL

This command specifies the logical record length to be used for a new output file. It does not override an existing record length that is specified in JCL or for a dataset that already exists.

–OUTFILE_LRECL(<length>)

length - This is an 8-character field specifying the logical record length.

00000080 - Eighty is the default record length.

–OUTFILE_MGMTCLASS

Synonyms Include: –OUTMCLASS

This command pertains to DF/SMS allocation of new files when doing PKUNZIP processing. If you specify these classes, they will be passed to DF/SMS when dataset allocation occurs.

–OUTFILE_MGMTCLASS(<SMS Management Class>)

See IBM's DF/SMS manuals for further information about this parameter.

–OUTFILE_OVERWRITE

Synonyms Include: –OVERWRITE, –NOOVERWRITE

The –OUTFILE_OVERWRITE command is used to update an existing file or member within a PDS.

–OUTFILE_OVERWRITE(Y|N)

Y - YES - The newly extracted dataset will overwrite the data in an existing dataset of the same name.

N - NO - The new dataset will not overwrite an existing dataset and the process will fail with an error message.

See –INSERT_MEMBER to add a dataset to an existing PDS.

–OUTFILE_PDS_ENQ

Synonyms Include: N/A

The –OUTFILE_PDS_ENQ command governs the level of disposition that will be used for a PDS or PDSE when processing an EXTRACT request. This affects both the EXTRACT job and other users in the system who have an existing PDS/PDSE open.

–OUTFILE_PDS_ENQ(OLD|SHR)

OLD - Specifies that a DISP=OLD be used.

SHR - Specifies that a DISP=SHR be used.

The greatest level of integrity is reached when jobs use DISP=OLD at the dataset level. However, when PDS datasets or PDSE Libraries are held open in long running jobs (such as an online system), it is not possible to use DISP=OLD in the PKZIP program to update a member.

DISP=SHR will result in the PKZIP program processing the PDS directory and its members without full dataset serialization. However, some level of protection is provided as follows:

1. During an EXTRACT process, PKUNZIP will test for an SPFEDIT ENQ on the PDS/PDSE member. If one exists in the system, then that member will be bypassed.

2. The operating system will provide protection for jobs using DISP=OLD:

   When another job holds the dataset with DISP=OLD, the PKZIP program will fail to obtain an allocation to the dataset.

   If the PKZIP program is updating the dataset and another job starts with DISP=OLD in its JCL, that job will wait until the PKZIP program closes and frees the file.

   If the PKZIP program is updating the dataset and another job or user attempts a dynamic allocation with DISP=OLD, that allocation request will fail.

3. The operating system may provide update protection for two different jobs attempting DISP=SHR updates. For example:

   If an IEBCOPY update is being performed against a PDS with DISP=SHR and PKZIP is running with –OUTFILE_PDS_ENQ(SHR), the PKZIP program may experience a system abend 213-30 when attempting to open the PDS directory. This is the way the system provides PDS directory integrity.

   Likewise, if the PKZIP program already has the PDS/PDSE open for output, the same IEBCOPY step in the other job would receive the 213-30 abend.

–OUTFILE_RECFM

Synonyms Include: –OUTTYPE

The –OUTFILE_RECFM command specifies the record format of the records in a newly extracted dataset. If not specified, the information is taken from the attributes stored in the ZIP archive.

–OUTFILE_RECFM(U|F|FA|FB|FBA|FBM|FBS|FM|V|VA|VB|VBA|VBM|VM)

U - Undefined records.

F - Fixed records.

FA - Fixed records with ISO/ANSI control characters.

FB - Fixed-Block records (note also that this default is ignored if an associated SMS command of –OUTFILE_DATACLASS is used).

FBA - Fixed-Block records with ISO/ANSI control characters.

FBM - Fixed-Block records with Machine control characters.

FBS - Fixed-Block Standard records.

FM - Fixed records with Machine control characters.

V - Variable records.

VA - Variable records with ISO/ANSI control characters.

VB - Variable-Block records.

VBA - Variable-Block records with ISO/ANSI control characters.

VBM - Variable-Block records with Machine control characters.

VM - Variable records with Machine control characters.

An undefined specification (U) will cause any –OUTFILE_LRECL specifications to be ignored. Similarly, any of the unblocked specifications will cause –OUTFILE_BLKSIZE specifications to be ignored.

–OUTFILE_SPACE_PRIMARY

Synonyms Include: –OUTPRIMARY

This command specifies the number of allocation units in the primary extent to be allocated to a newly extracted dataset.

The default is not used if –OUTFILE_DATACLASS is specified.

```
–OUTFILE_SPACE_PRIMARY(<allocation units>)
```

allocation units - This an 8-character field specifying the number of allocation units for the primary extent allocation.

00000010 - Ten is the default.

–OUTFILE_SPACE_RLSE

Synonyms Include: –OUTFILE_RLSE, –OUTFILE_RELEASE, –OUTFILE_SPACE_RELEASE, –OUTRLSE, –OUTNORLSE

This command indicates that when a new file is closed using PKUNZIP processing, additional cylinders or tracks should be released from the allocation.

```
–OUTFILE_SPACE_RLSE(Y|N)
```

Y - YES - The deallocated free space is released following compression. **This is the default action taken for sequential datasets**.

N - NO - The deallocated free space is not released following compression. **This is the default action taken for partitioned datasets** (since the extra space may be needed by other members within the same PDS).

–OUTFILE_SPACE_SECONDARY

Synonyms Include: –OUTSECONDARY

This command specifies the number of allocation units in the secondary extent to be allocated to a newly extracted dataset.

The default is not used if –OUTFILE_DATACLASS is specified.

```
–OUTFILE_SPACE_SECONDARY(<allocation units>)
```

allocation units - This an 8-character field specifying the number of allocation units for the secondary extent allocation.

0000010 - Ten is the default.

–OUTFILE_SPACE_TYPE

Synonyms Include: –OUTSPACE

This command specifies the type of allocation units that are used at the allocation of a newly extracted dataset. The allocation units may be one of five choices with CYL (cylinders) as the default. Note that the default is not used when –OUTFILE_DATACLASS is specified.

–OUTFILE_SPACE_TYPE(<TRK|CYL|BLK|MB|KB>)

TRK - (also TRKS and TRACKS) Allocation by tracks.

CYL - (also CYLS and CYLINDERS) Allocation by cylinders.

BLK - (also BLKS and BLOCKS) Allocation by blocks (Note that the block size is specified in the –OUTFILE_BLKSIZE command.)

KB - (also KILOBYTES) Allocation by Kilobytes for the ICF catalog environment only.

MB - (also MEGABYTES) Allocation by Megabytes for the ICF catalog environment only.

Note: Both the primary and secondary extents are allocated at 10 allocation units unless changed by the –VSAM_SPACE_PRIMARY or the –VSAM_SPACE_SECONDARY commands.

This command specification can be overridden at the data level by the –VSAM_DATA_SPACE_TYPE command. At the data level, the corresponding cluster information is not recognized.

–OUTFILE_STORCLASS

Synonyms Include: –OUTSCLASS

This command pertains to DF/SMS allocation of new files when doing PKUNZIP processing. If you specify these classes, they will be passed to DF/SMS when dataset allocation occurs.

–OUTFILE_STORCLASS(<SMS Storage Class>)

See IBM's DF/SMS manuals for further information about this parameter.

–OUTFILE_UNIT

Synonyms Include: –OUTUNIT

For a newly extracted dataset, the generic units for the archive can be specified using the –OUTFILE_UNIT command.

–OUTFILE_UNIT(<units>)

unitname - This is an 8-character field specifying the name of the generic unit to which the output dataset is to be allocated.

SYSDA - This is the default specification.

–OUTFILE_VOLUMES

Synonyms Include: –OUTVOL

For a newly extracted dataset, the volume(s) is specified using the –OUTFILE_VOLUMES command.

–OUTFILE_VOLUMES(<volname>[<volname> <volname>…])

volname - A 217-byte field specifying the names of volume(s) (separated by blanks) onto which a newly extracted dataset is allocated. There may be up to 31 volume names specified with this command.

For an output that is a new member of a new PDS, the first <volname> will only be used.

For a VSAM file, the volumes are specified at the Cluster Level.

–PAD_CHAR

Synonyms Include: –PAD

When extracting data into fixed length records, specify the pad character with the command –PAD_CHAR. If the command is not specified and padding is needed, the default will be spaces (X'40') for TEXT and nulls (X'00') for BINARY extraction.

```
–PAD_CHAR(<pad char>)
```

pad char - May be one of the following:

None - For –PAD_CHAR(), the space (X'40') is used.

Any EBCDIC character.

Any Hexadecimal character with the format X('<hex character>').

Multiple pad characters will be used if needed to fill in at the end of the record to make it the required fixed record length.

–PAD_VSAM

Synonyms Include: –PADVSAM, –NOPADVSAM

This command instructs the PKUNZIP program to pad variable length records with a character(s) specified by the –PAD_CHAR command to the length specified in the –VSAM_RECORDSIZE command (average and maximum lengths must be the same).

```
–PAD_VSAM(Y|N)
```

Y - YES - Records are padded with the pad character specified in –PAD_CHAR. If the lengths specified in the –VSAM_RECORDSIZE command are of different lengths, padding will not occur.

N - NO - Records are not padded.

–PARMLIB_DSNAME_UNZIP

Synonyms Include: –UNZIPCONFIG

```
–PARMLIB_DSNAME_UNZIP(<dataset>)
```

dataset - ***PKZIP MVS™*** can be configured or customized to operate in a number of ways. The name of the dataset containing the configuration specifications for UNZIP processing is specified by the use of this command. The default command for this dataset is –NULLFILE.

Note that some installations try to eliminate any allocation of a PARMLIB or CONFIG dataset through –PARMLIB_DSNAME_ZIP and –PARMLIB_DSNAME_UNZIP.

If no installation-supplied dataset commands are desired, then ACZDFLT parameters may be set to bypass the allocation attempt. Only //SYSIN DD and EXEC PARM='...' parameters will be processed.

–PARMLIB_DSNAME_ZIP

Synonyms Include: –ZIPCONFIG

–PARMLIB_DSNAME_ZIP(<dataset>)

dataset - ***PKZIP MVS™*** can be configured or customized to operate in a number of ways. The name of the dataset containing the configuration specifications for ZIP processing is specified by the use of this command. The default command for this dataset is –NULLFILE.

–PARMLIB_FILE_WAIT_MAX

Synonyms Include: N/A

If the file indicated by –PARMLIB_DSNAME_ZIP or –PARMLIB_DSNAME_UNZIP is in use elsewhere and cannot be opened, the command –PARMLIB_FILE_WAIT_MAX indicates the maximum amount of time ***PKZIP for MVS™*** will wait for the file to become available before abnormally ending the job. The default setting is five minutes.

–PARMLIB_FILE_WAIT_MAX(<HHMMSSTH>)

HHMMSSTH:

HH - Hours.
MM - Minutes.
SS - Seconds.
T - Tenths of a second.
H - Hundredths of a second.

00050000:

5 minutes is the default.

–PARMLIB_FILE_WAIT_TIMER

Synonyms Include: N/A

If the file indicated by –PARMLIB_DSNAME_ZIP or –PARMLIB_DSNAME_UNZIP is in use elsewhere and cannot be opened, the command –PARMLIB_FILE_WAIT_TIMER is the polling time used during the wait process. The default setting is five seconds.

–PARMLIB_FILE_WAIT_TIMER(<HHMMSSTH>)

HHMMSSTH:

HH - Hours.
MM - Minutes.
SS - Seconds.
T - Tenths of a second.
H - Hundredths of a second.

00000500:

5 seconds is the default.

–PASSWORD

Synonyms Include: –PASS, –PWD

To encrypt a ZIP archive file, the –PASSWORD command is used to establish an associated password for that file.

–PASSWORD(<userpw>)

userpw - This is your selected password needed for later decrypting the ZIP archive file. The encrypted file will need this password for access for extraction.

The password:

Is case-sensitive - Capital and lower case letters should be just that. The following passwords are considered three different passwords: Password, PASSWORD, or password. If the password is being input from JCL, take note that the JCL editor may capitalize all the letters of the password.

May be 1-200 characters in length.

This is not stored in the ZIP archive and, as a result, care must be taken to keep passwords secure and accessible by some other source. Different passwords may be used for various files within a ZIP archive, although only one password may be specified per run.

Password translation is done from EBCDIC to ASCII using the –TRANSLATE_TABLE_FILEINFO. When cross-platform exchanges are done with encrypted Archives, care should be taken to use characters that will be acceptable to both platforms with the translate table in use.

–PATH

Synonyms Include: –NOPATH

The –PATH command determines how an MVS filename is converted to a ZIP archive format.

–PATH(Y|N)

When using –PATH(Y)

When converting a filename from MVS format to ZIP archive format, the –PATH(Y) command is specified so that all of the dataset levels are used in the archive name. –PATH(Y) is the default.

Example:	
Given the PDS member:	PROJECT.DEPT.SOURCE(TEST)
and a command of:	–PATH(Y)
the ZIP internal filename will be:	PROJECT/DEPT/SOURCE/TEST

Example:	
Given the PDS dataset that contains member CLOCK00	SYS1.PARMLIB
and a command of:	–PATH(Y)
the ZIP internal filename for that member will be:	SYS1/PARMLIB/CLOCK00

When using –PATH(N)

When converting a filename from MVS format to ZIP archive format, the –PATH(N) command is specified so that the last level of the dataset is used as the archive name. This command is not used if a matching –ZIPPED_DSN command exists.

Example:	
Given the PDS member:	SYS1.PARMLIB(CLOCK00)
and a command of:	–PATH(N)
the ZIP internal filename will be:	CLOCK00

–PKSUPPRC

Synonyms Include: N/A

–PKSUPPRC is a non-default command that allows the return code to be suppressed for the following message:

ZPAM092E - Nothing to do.

ZPAM093W - No files match; Initializing/Copying Archive.

ZPEX013W - Truncation.

–PRESERVE_CMD_SPACES

Synonyms Include: N/A

In releases of PKZIP 2.61 and prior, a " |" was required to identify a command continuation, a blank preceding the " |" was needed to identify the continuation action. The support of continuation command records with imbedded blanks that was added with ***PKZIP MVS™*** 5.5 (for extended filenames) required all occurrences of " |" (preceding space) to be changed to "|".

–PRESERVE_CMD_SPACES

Y - YES - will be the default and will be required for the preservation of preceding spaces when required for specific command values, for example, UNIX-format file names with imbedded blanks and –ARCHIVE_COMMENT text.

N - NO - backward-compatibility command will provide you with the flexibility to remove blanks preceding the "|" as in earlier releases of ***PKZIP MVS™*** (provided with fix TT1053).

Warning: Space preservation for current and future commands is predicated on the default –PRESERVE_CMD_SPACES=Y. Control cards should be converted to the ***PKZIP for MVS™*** 5.x format (no blanks preceding "|" for continued lines).

–PROCESS_ALIAS

Synonyms Include: –ALIASMEMBER, –NOALIASMEMBER

–PROCESS_ALIAS(Y|N)

During ZIP processing, the –PROCESS_ALIAS(Y) command specifies that the alias entries for selected PDS members are to be retained for the real member. These stored attributes then may be used when extracting the file to a PDS.

During UNZIP processing, the –PROCESS_ALIAS(Y) command specifies that saved alias entries for selected PDS members are to be restored to the PDS directoy in association with the real member.

Processing Notes:

- Alias members are not selectable as members or files from the Archive. The “real” member must be selected.
- –SAVE_FILE_ATTRIBUTES(CENTRAL) must be active during the ZIP process for this command to take effect.

–RECALL_TO_ZIP

Synonyms Include: –RECALL, –NORECALL, –SELECT_MIGRATED

This command instructs ***PKZIP for MVS™*** to either recall a dataset with DFHSM or to bypass that dataset if a recall is required. This will speed up processing if migrated datasets are not required to be zipped. The catalog information is reviewed for volume serial (MIGRATE or ARCHIVE) to identify datasets which are migrated. (“ARCHIVE” is used by some non-IBM storage management products).

–RECALL_TO_ZIP(Y|N)

Y - YES - Recall a dataset using DFHSM. Note that this specification may incur significant processing delays as DFHSM performs the recalls so that file attributes can be checked. File attributes must be checked to insure that Partition Dataset information is all specified and characterized before file selection can occur.

N - NO - Bypass recall of DFHSM datasets.

–RECURSE_LEVELS

Synonyms Include: –RECURSE, –NORECURSE

During ZIP file selection with masking, this specifies whether or not to use file names represented by wildcard specifications.

-RECURSE_LEVELS(Y|N)

When specifying Y (YES):

Additional dataset levels below the qualifiers specified are included in the match for a user specified dataset.

Example:	
For the selection:	XXX.YYY(*)
and a command of:	-RECURSE_LEVELS(Y)
the following datasets would be found:	XXX.YYY XXX.YYY.ZZZ XXX.YYY.REF XXX.YYY.Z12

or

Example:	
For the selection:	PAYROLL.DEPT(*)
and a command of:	-RECURSE_LEVELS(Y)
the following datasets would be found:	PAYROLL.DEPT.ENG PAYROLL.DEPT.ACC05 PAYROLL.DEPT.MKT PAYROLL.DEPT.ADVERT

When specifying N (NO):

Only the specified dataset levels from your file selection are used in the match for file selection.

Example:	
For the selection:	XXX.YYY(*)
and a command of:	-RECURSE_LEVELS(N)
the only dataset found would be:	XXX.YYY
and would not be:	XXX.YYY.ZZZ XXX.YYY.REF XXX.YYY.Z12

or

Example:	
For the selection:	PAYROLL.DEPT(*)
and a command of:	-RECURSE_LEVELS(N)
the only dataset found would be:	PAYROLL.DEPT
and would not be:	PAYROLL.DEPT.ENG PAYROLL.DEPT.ACC05 PAYROLL.DEPT.MKT PAYROLL.DEPT.ADVERT

–SAVE_FILE_ATTRIBUTES

Synonyms Include: –ATTRIBCENTRAL, –ATTRIBLOCAL, –ATTRIB, –NOATTRIB, USE_FILE_ATTRIBUTES

☺ - Cross Platform Compatible command (VSE and OS/400).

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –SAVE_FILE_ATTRIBUTES command specifies whether and where to save the attributes of this compressed file.

–SAVE_FILE_ATTRIBUTES(CENTRAL|LOCAL|BOTH|NONE|NO|N)

When used in PKZIP program processing:

CENTRAL - Allocation attributes for the ZIPPED file are stored in the Central Directory. ***PKZIP MVS™*** uses this information when extracting a file.

> **Note: The –OUTFILE series of commands may be used during PKUNZIP processing to ignore stored attributes.**

LOCAL - Attributes for the ZIP file are stored (only) in the Local Directory. ***PKZIP MVS™*** does not use the Local Directory when extracting a file.

BOTH - The attributes for a compressed file are to be stored in the ZIP archive in both the Central and the Local Directories of an archive when the BOTH modifier is issued.

NONE|NO|N - The attributes for a compressed file are not to be stored in the ZIP archive. This is useful when the Archive is to be sent to another platform where the allocation information is not referenced. It also serves to reduce the size of the Archive.

When used in PKUNZIP program processing:

NONE|NO|N - The attributes for a ZIP file should not be used when creating a new extracted dataset. Instead, the –OUTFILE series of commands may specify attributes for the new extracted dataset. Any other value will cause ***PKZIP MVS™*** to blend the extended attributes saved in the Archive with override commands for new dynamically allocated files.

The attributes are not used when an OUTFILE_DDNAME JCL allocation is used. The user should specify all appropriate values through JCL or pre-allocation.

–SAVE_LRECL

Synonyms Include: –RDW, –USE_SAVED_LRECL, –ZDW

☺ - This command is comaptible with VSE.

☹ - This command is not compatible with UNIX, MVS, OS/400, and Windows.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command is used in combination with –DATA_TYPE(BINARY) during ZIP processing to specify record lengths should be retained with the ZIPPED file. This is particularly useful for files containing variable-length records that need to be restored to their original length during UnZip processing.

–SAVE_LRECL(Y|N)

Y - YES - Specifies that record length information is to be included in the Zip archive.

N - NO - Specifies that record length information is not to be included.

It is highly recommended that VSAM files ZIPPED as BINARY should have –SAVE_LRECL(Y) specified even if the catalog indicates the average and maximum recordsize to be the same. This is not a guarantee that all records in the VSAM CLUSTER are of the same length.

–SAVE_LRECL(Y) should always be specified with Load modules.

It may be the case that a particular platform does not support the –SAVE_LRECL command and does not use stored record lengths in a binary file and therefore the file should be processed as straight –DATA_TYPE(BINARY) , otherwise formatting problems may be encountered with the data.

This command does not apply to files ZIPPED as TEXT.

–SELECT_CATALOGED_ALIAS

Synonyms Include: –ALIAS_NAME, –NOALIAS_NAME, –SELECT_DSN_ALIAS

This parameter specifies whether ALIAS catalog entries are to be eligible for processing when performing a ZIP request for –ACTION(ADD) or –ACTION(UPDATE).

–SELECT_CATALOGED_ALIAS(Y|N)

Y - YES - alias catalog entries will be processed.

N - NO - alias catalog entries will not be processed.

This command specifies that if there is a dataset named XYZ that has an alias defined as ABC, ***PKZIP for MVS™*** processing will zip the XYZ dataset if ABC is requested. It is an alternative way of asking for files.

–SELECT_FROM_PDS

Synonyms Include: –PDS_TARGET, –ZIPCUR

When used in PKZIP program processing:

The –SELECT_FROM_PDS command is used as a shortcut to specify the current higher level components which would apply to the files that follow in the command list. It eliminates having to enter the higher level dataset components each time a different dataset is referenced.

–SELECT_FROM_PDS(<PDS name>)

Example:	
Zipping:	ABC
with commands of:	–SELECT_FROM_PDS(DOG.PONY.SHOW) ABC
will select the file for zipping:	DOG.PONY.SHOW(ABC)

When used in PKUNZIP program processing:

The –SELECT_FROM_PDS command is used to designate an output library for files to be extracted into. It is commonly used when a PDS is not specified in a dataset name, for example, the name levels were dropped by the –HIERARCHY(N) command during ZIP processing when the Archive was created.

–SELECT_FROM_PDS (<PDS name>)

Example:	
Unzipping:	ABC
with a command of:	–SELECT_FROM_PDS (DOG.PONY.SHOW)
will extract the PDS member:	DOG.PONY.SHOW(ABC)

See –UNZIPPED_DSN to specify high level qualifiers in a more general fashion.

–SELECT_TAPE

Synonyms Include: –NOTAPE

This command specifies whether tape files are to be processed when requesting datasets for ZIP processing via the catalog.

–SELECT_TAPE(Y|N)

Y - YES - All tape files in the catalog will be processed.

N - NO - Tape files will be filtered out during processing of the catalog.

See also: –SELECT_VSAM, –SELECT_MIGRATED, and –SELECT_GDGALL.

–SET_ERROR_RC

Synonyms Include: N/A

The –SET_ERROR_RC may be used to set a firm return code when an error has been detected. Internal return codes of 8 or above will be converted to this value. This optional feature may be of use to installations converting from PKZIP for MVS 2.x, which uses RC=24 for severe errors.

–SET_ERROR_RC(<nbr>)

nbr - Return code to be passed to the system.

–SHOW_SETTINGS

Synonyms Include: –SS

This command which causes current command settings to be displayed in the output at the point in the input that the –SHOW_SETTINGS command is invoked. Since command settings may come from the Execute Parm, the Parmlib Configuration File, or from SYSIN, the use of the –SHOW_SETTINGS command is useful in showing the combined effect of all sources leading up to the request.

–SHOW_SETTINGS

No parameter is required.

> **Note: This command does not override or interrupt the processing request in effect for the run(–ACTION). If a standalone report is desired without attempting ZIP/UNZIP Archive processing, use the command sequence "–SS –PATCH_REPORT".**

–SIMULATE

Synonyms Include: N/A

This command runs file selection processes for –ACTION(ADD), –ACTION(EXTRACT), –ACTION(FRESHEN), and –ACTION(UPDATE), but does not perform actual data manipulations for the files selected or for the output archive. Compression and Decompression algorithms will be bypassed. The input archive will be opened and read for directory information. –STAGE_TAPE_ON_DISK will also be acted upon when specified or required.

–SIMULATE(Y|N)

Y - YES - Simulation of the file selection processes will occur.

N - NO - Full processing file processing will occur.

Note: This command is helpful when learning to code the –ZIPPED_DSN and –UNZIPPED_DSN commands.

–SNAP_SYSOUT_CLASS

Synonyms Include: N/A

This command specifies the SYSOUT class to be used for SNAP dumps. This feature is used only in conjunction with diagnostic features of ***PKZIP for MVS™*** and may not necessarily be used by an end user of the product.

```
–SNAP_SYSOUT_CLASS(<class>)
```

class - A one-character class assigned for the output of a SNAP dump.

*** -** is the default.

–STAGE_TAPE_ON_DISK

Synonyms Include: –STAGE_TAPE_TO_DISK

This command specifies that input from a sequential device be stored in a temporary dataset.

```
–STAGE_TAPE_ON_DISK(Y|N)
```

Y - Yes - Processing occurs on disk rather than on tape.

N - No - Processing occurs on tape, thus incurring significant processing degradation.

When reading a cartridge-based archive, the input can be stored in a temporary dataset with the –STAGE_TAPE_ON_DISK command. This occurs automatically when reading a 3420 (reel-to-reel) archive.

Should allocated temporary space be insufficient, the temporary dataset is not used and processing continues with the tape. Note that this will have an impact on elapsed processing time.

It is helpful to include FREE=CLOSE in the DD statement in the JCL. This frees up the tape once the copy of the data has been made. If it is not included, the tape must remain mounted for the duration of ZIP processing.

> **Warning:** If an //ARCHTEMP DD is found in the JCL, it will be over-written with the input archive. This DDNAME should not be used in a ***PKZIP MVS™*** job-step for any other purpose.

–STRIP_CHAR

Synonyms Include: –STRIP

This command specifies an ending character to be removed from the end of each record before it is compressed. There is no default as this process does not occur unless specified.

```
-STRIP_CHAR(<strip char>)
```

strip char - This is a single entry for the character(s) to be removed from the end of each record before compressing. One of three types may be entered:

No character specifies that trailing spaces (hex ’40’) are removed from every record.

Any EBCDIC character.

Any Hexadecimal character in the format: STRIP_CHAR(X'7B').

If multiple characters occur at the end of the record, all occurrences of the character are removed.

Use caution with this command as the dataset will be modified.

–SUPPRESS_DYNALLOC_MSGS

Synonyms Include: –NODYNMSGS

This command specifies that the dynamic allocation messages that appear in the job log be suppressed. This will not affect severe errors.

–SUPPRESS_DYNALLOC_MSGS

PKZIP for MVS™ performs dynamic allocation requests for various files (archive, parameter, input, output, and temporary). During the system-service requests, the operating system may attempt to issue messages to the joblog or foreground TSO session screen. These messages are classified by level, ranging from Informational to Error conditions. ***PKZIP for MVS™*** intercepts many of the Dynamic Allocation return code conditions and provides its own reporting according to the dataset request being performed.

–SUPPRESS_DYNALLOC is the default, which limits the operating system to reporting "Error" conditions (as the operating system defines "error").

If additional dynamic allocation information is needed for problem determination purposes, the ***PKZIP for MVS™*** technical support staff will provide additional commands that will provide tracing of Dynamic Allocation activities.

Note: An ACZDFLT setting of –TRACE_DYNALLOC=0 can be used to make this the default.

–SYSPRINT_DCB

Synonyms Include: N/A

This command can only be entered through ACZDFLT or as an EXEC parameter. The SYSPRINT DCB attributes can be customized.

–SYSPRINT_DCB(FB132|FBA121|FBA133|FA121)

FB132 - SYSPRINT DCB attributes will be fixed block 132 no ASA.

FBA121 - SYSPRINT DCB attributes will be fixed block 121 with ASA.

FBA133 - SYSPRINT DCB attributes will be fixed block 133 with ASA.

FA121 - SYSPRINT DCB attributes will be fixed 121 with ASA.

The ASA control character (character in column 1) will be a blank.

–SYSPRINT_SYSOUT_CLASS

Synonyms Include: N/A

This command specifies the SYSOUT class to be used for SYSPRINT messages when a SYSPRINT allocation is not provided for the job/session.

–SYSPRINT_SYSOUT_CLASS(<class>)

class - A one-character class assigned for the output of a SYSPRINT listings.

The default is the JCL MSGCLASS associated with the runtime environment.

Record Length: 132.

Format: FB.

–TEMP_BLKSIZE

Synonyms Include: –TEMPBLKSIZ

This command specifies the block size of a temporary ***PKZIP for MVS™*** dataset.

–TEMP_BLKSIZE(DYNAMIC|SMS|value)

DYNAMIC/SMS - A dynamically computed value will be requested by ***PKZIP MVS™*** (although SMS or allocation routines in the operating system may override the value).

Value - A blocksize value; recommended to be sized at half-track for the selected –TEMP_UNIT.

When either DYNAMIC or SMS is specified, a dynamically computed value will be requested by ***PKZIP MVS™*** (although SMS or allocation routines in the operating system may override the value).

–TEMP_DATACLASS

Synonyms Include: –TEMPDCLASS

–TEMP_DATACLASS(<data class>)

Use this command to specify or override value for temporary work File allocation requests in a DF/SMS - controlled environment.

data class - Specifies the DF/SMS data class receiving the temporary ZIP dataset.

See IBM's DF/SMS manuals for further information about this parameter.

–TEMP_MGMTCLASS

Synonyms Include: N/A

–TEMP_MGMTCLASS(<mgmt class>)

Use this command to specify or override value for temporary work File allocation requests in a DF/SMS - controlled environment.

mgmt class - Specifies the DF/SMS management class receiving the temporary ZIP dataset.

See IBM's DF/SMS manuals for further information about this parameter.

–TEMP_RECFM

Synonyms Include: –TEMPTYPE

–TEMP_RECFM(U|F|FB)

The command specifies the record format of a temporary PKZIP MVS dataset.

U - Undefined record format.

F - Fixed record format.

FB - Fixed block record format.

–TEMP_SPACE_PRIMARY

Synonyms Include: –TEMPPRI, –TEMPPRIMARY

```
–TEMP_SPACE_PRIMARY(<allocation units>)
```

allocation units - Specifies the number of allocation units for the primary extent of the temporary ZIP dataset.

Default is the same as –ARCHIVE_SPACE_PRIMARY.

–TEMP_SPACE_SECONDARY

Synonyms Include: –TEMPSEC, –TEMPSECONDARY

```
–TEMP_SPACE_SECONDARY(<allocation units>)
```

allocation units - This is the size of the secondary extent in allocation units for the temporary ZIP dataset.

Default is the same as –ARCHIVE_SPACE_SECONDARY.

–TEMP_SPACE_TYPE

Synonyms Include: –TEMPSPACE

```
–TEMP_SPACE_TYPE(TRK|CYL|BLK)
```

TRK - Tracks.

CYL - Cylinders.

BLK - Blocks (with the size specified in the –TEMP_BLKSIZE command).

–TEMP_STORCLASS

Synonyms Include: –TEMPSCLASS

```
–TEMP_STORCLASS(<storclass>)
```

Use this command to specify or override value for temporary work File allocation requests in a DF/SMS - controlled environment.

storclass - This is the DF/SMS storage class requested in placing the temporary ZIP dataset. An installation's DF/SMS ACS routine may reset the value.

See IBM's DF/SMS manuals for further information about this parameter.

–TEMP_UNIT

Synonyms Include: –TEMPUNIT

```
–TEMP_UNIT(<unit name>)
```

unit name - Specifies the generic unit name indicating where the dataset is to be allocated. SYSDA is the default if not provided.

Use the –SHOW_SETTINGS command to determine the installation's selected default values.

Note: The defaults may not reflect the installation values by the product installer.

–TEMP_VOLUMES

Synonyms Include: –TEMPVOL

```
-TEMP_VOLUME(<volname>[ <volname> <volname> …..]
```

volname - Specifies 1 to 31 volumes that indicate where the temporary ZIP dataset is to be allocated. Separate multiple <volume name>s by spaces.

This command is used in conjunction with –TEMP_UNIT to direct work files to a specific location. PKZIP MVS will use values specified in its dynamic allocation request. The installation's storage management controls may redirect the actual file location.

Note: The defaults may not reflect the installation values by the product installer.

–TRACE_TABLE_SIZE

Synonyms Include: N/A

This command specifies the size of the internal trace table.

```
–TRACE_TABLE_SIZE(<tabsize>)
```

tabsize - This is an 8-byte field containing the size of the trace table.

–TRANSLATE_TABLE_DATA

Synonyms Include: –TRAN

☺ - Cross Platform Compatible command (VSE, MVS, OS/400, UNIX, and Windows).

☠ - Be aware that if this command is used incorrectly, you could incur problems.

```
–TRANSLATE_TABLE_DATA(<translation table name>)
```

When used in PKZIP program processing:

Use the –TRANSLATE_TABLE_DATA command to identify a particular translation table to be used when converting text file data from one character set to another. This command would be used, for example, when converting a file from EBCDIC to ASCII, which is the standard the ZIP Archive text format.

Where **<translation table name>** specifies a useable table name for translation. ASCIIUS is the default if –TRANSLATE_TABLE_DATA is not specified. This is the table specified in the defaults module and it can be changed by customizing the ACZDFLT module.

When used in PKUNZIP program processing:

Use the –TRANSLATE_TABLE_DATA command to identify a particular translation table to be used when extracting a text file from ASCII to anther character set. This command would be used, for example, when converting a ZIP archive file in ASCII to a non-MVS format of EBCDIC.

–TRANSLATE_TABLE_DATA(<translation table name>) where <translation table name> specifies a useable table name for translation.

ASCIIUS is the default if –TRANSLATE_TABLE_DATA is not specified. The default table selection may be changed to a different table.

PKZIP for MVS™ provides certain "ready to use" translation tables commonly used in an OS/390 environment. These tables are provided "as is" and are not supported as part of ***PKZIP for MVS***™. It is your responsibility to ensure that data translation mapping satisfies their requirements. More information can be found in the FAQ at http://www.pkzip.com.

Language	EBCDIC Code Page	ASCII Code Page	EURO/ ASCII Code Page	EBCDIC Code Set ID	ASCII Code Set ID	EURO/ ASCII CODE Set ID	Table Name ASCII	Table Name EURO
German	273	850	858	EB	AA	AI	TRTEBAA	TRTEBAI
Spanish	284	850	858	EJ	AA	AI	TRTEJAA	TRTEJAI
Portuguese	282	850	858	EI	AA	AI	TRTEIAA	TRTEIAI
Italian	280	850	858	EG	AA	AI	TRTEGAA	TRTEGAI
Daniish	277	850	858	EE	AA	AI	TRTEEAA	TRTEEAI
Norwegian	277	850	858	EE	AA	AI	TRTEEAA	TRTEEAI
Swedish	278	850	858	EF	AA	AI	TRTEFAA	TRTEFAI
Finnish	278	850	858	EF	AA	AI	TRTEFAA	TRTEFAI
French	297	850	858	EM	AA	AI	TRTEMAA	TRTEMAI

–TRANSLATE_TABLE_FILEINFO

Synonyms Include: –FTRAN

☺ - Cross Platform Compatible command (VSE, MVS, OS/400, UNIX, and Windows).

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –TRANSLATE_TABLE_FILEINFO command specifies a translation table to be used with file information such as comments, file names, andpassword usage for an encrypted file. The default is ASCIIUS if this command is not specified.

```
-TRANSLATE_TABLE_FILEINFO(<trantable>)
```

trantable - This is a name of a loadable translation table that is supplied with the product or customized by the installation.

Use this command when filenames are in an incompatible format with the target platform or when standard translation tables contain indecipherable characters from when the file was previously translated.

–UNZIPPED_DSN

Synonyms Include: –NOA, –HLQ, –UNZIPPED_DSNAME

One or more –UNZIPPED_DSN commands may be used to modify high level qualifiers when extracting files. During filename transformation (from Archive filename format to MVS dataset name format), matching archive file high level qualifiers are replaced with an MVS high level qualifier specified in this command. A generalized renaming process can be made by using wildcard specifications.

The basic format of the command is:

–UNZIPPED_DSN([<*Zipfile_path*>],[<*MVS_hlq*>])

Note: Either of the fields may be blank, but not both.

Note: In previous versions of PKZIP for MVS, the '/' character was used to separate the two parameters. This character is still supported, but the ',' is recommended as this is consistent with other commands and removes confusion about the use of the '/' character in the Zip file name.

The four possible functions performed by this command include:

High Level Replacement

–UNZIPPED_DSN(<*Zipfile_path*>,<*MVS_hlq*>)

Example:	
Given the archive:	MDB/TYPE/RATE
and a command of:	–UNZIPPED_DSN(MDB.TY,XXX.) (note delimiter in newname)
the result will be:	XXX.PE.RATE

Example:	
Given the archive:	MDB/TYPE/RATE
and a command of:	–UNZIPPED_DSN(*,XXX) (note use of wildcard for high level qualifier in oldname)
the result will be:	XXX.TYPE.RATE

Example:	
Given the archive:	MDB/TYPE/RATE
and a command of:	–UNZIPPED_DSN(MDB.?YPE.,XXX) (note delimiter in oldname)
the result will be:	XXX.RATE

High Level Prefixing

–UNZIPPED_DSN(,<*MVS_hlq*>)

Example:	
Given the archive:	MDB/TYPE/RATE
and a command of:	-UNZIPPED_DSN(,NEW.) (note delimiter in new name)
the result will be:	NEW.MDB.TYPE.RATE
or a command of:	-UNZIPPED_DSN(,NEW) (note no delimiter in new name)
the result will be:	NEWMDB.TYPE.RATE

High Level Removal

–UNZIPPED_DSN(<*Zipfile_path*>,)

Example:	
Given the archive:	MDB/TYPE/RATE
and a command of:	-UNZIPPED_DSN(M,)
the result will be:	DB.TYPE.RATE (with the "M" removed)

Complete Replacement

–UNZIPPED_DSN(**,<*MVS_hlq*>)

Example:	
Given the data:	MDB/TYPE/RATE
and a command of:	-UNZIPPED_DSN(**,NEW.VER.DATA)
the result will be:	NEW.VER.DATA

Parameter Usage:

- <*Zipfile_path*> defines the high-level qualifier characters of the input ZIP file name that are to be substituted by the <*MVS_HLQ*>. This value can be up to 80 characters long and may specify *wild* characters to assist in the matching. The *wild* characters that can be specified are:
    - "*" to match any number of characters (within one level).
    - "?" to match a single character (except a qualifier separation character).
- <MVS_hlq> specifies the characters that are to be used to replace those specified in the first operand (if any) and prefixed to the remainder of the Archive filename. A maximum of 54 characters may be specified and should match MVS dataset naming conventions.

    '*'

- Processing Notes.

- If you are uncertain about the results that will be achieved by the use of –UNZIPPED_DSN, it is recommended that trial runs be performed with the –SIMULATE command. This will cause ***PKZIP MVS***™ to issue standard extraction messages that contain the target DSN values without actually extracting the files. This avoids excessive processing time and errant dataset creation when undesired filename results are experienced.

- The –UNZIPPED_DSN command is not recommended when using –NOHIERARCHY, -OUTDD, or –ZIPCUR commands, as these commands can also change the output dataset name used, in potentially conflicting ways.

- The –UNZIPPED_DSN command is processed after the –FILE_EXTENSION command has been used. –FILE_EXTENSION(DROP) causes the removal of the 'extension' in the ZIP file name, in which case the extension should not be used when specifying the *Zipfile_path*.

- When attempting to extract files to PDS members, the command –OUTFILE_DSNTYPE (PDS I PDSE) may be used in combination with this command. In addition, by specifying a PDS member mask in *newname*, a PDS target will be assumed. For example: –UNZIPPED_DSNAME (**,MY.NEW.PDS(*)).

- Message ZPAM183E will be issued when the target MVS name is determined to fail MVS naming conventions (such as when the resulting filename is too long for the target dataset type, or DSN qualifiers are not properly constructed with period separators).

  - ZPAM91I GENERATED MVS DSN LEVEL TOO LONG MAS.IR4006DZMTVT.

  - ZPAM183E –UNZIPPED_DSN(…/ parm2) Name is invalid.

- The input –UNZIPPED_DSN commands are searched in the order specified until a match is found with the beginning of the ZIP archive file name. Although many commands may be specified to account for various filename matches, one and only one is used to resolve the MVS_hlq once a match is found.

–VSAM

Synonyms Include: –NOVSAM, –SELECT_VSAM

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

To access or not access VSAM files during wildcard selections the –VSAM command is specified. This only occurs for wildcard cases.

–VSAM (Y | N)

Y - YES - Any VSAM files that are used in multiple dataset selections are included when using a wildcard request.

N - NO - The VSAM file(s) within a file selection are ignored when the selection contains a wildcard. If no wildcard is used in the selection, the VSAM file is used regardless.

Note that all–VSAM commands use the Access Methods Services IDCAMS utility to help define a new (or update an existing) data component, for a VSAM Cluster containing a ZIP Archive. See the Access Methods Services manual for specific information on use of this parameter.

–VSAM_ACCOUNT

Synonyms Include: N/A

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_ACCOUNT parameter defines the accounting information to be provided to Access Methods Services during a DEFINE CLUSTER.

The IDCAMS equivalent for this command is ACCOUNT(accounting information).

```
-VSAM_ACCOUNT(<acctinfo>)
```

acctinfo - A 32-character field containing accounting information.

–VSAM_ATTEMPTS

Synonyms Include: –OUTATTEMPTS, –OUTDATAATT

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_ATTEMPTS parameter defines the number of password attempts that are permitted to Access Methods Services during a DEFINE CLUSTER.

The IDCAMS equivalent for this command is ATTEMPTS(number).

```
–VSAM_ATTEMPTS(<number>)
```

number - The number of attempts that will be allowed at the console in response to a prompting message.

–VSAM_AUTH_EP

Synonyms Include: –OUTAUTH, –OUTDATAAUTH

The –VSAM_AUTH_EP parameter supplies the entry point of a user security verification routine to Access Methods Services during a DEFINE CLUSTER.

The IDCAMS equivalent for this command is AUTHORIZATION(entrypoint).

```
–VSAM_AUTH_EP(<entry point>)
```

entry point - The entry point name of your security verification routine.

See also –VSAM_AUTH_STRING below.

–VSAM_AUTH_STRING

Synonyms Include: –OUTASTR, –OUTDATAASTR

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_AUTH_STRING parameter supplies a string of information to be passed to your security verification routine to Access Methods Services during a DEFINE CLUSTER.

The IDCAMS equivalent for this command is AUTHORIZATION(entrypoint string).

–VSAM_AUTH_STRING(<string>)

string - The string of information to be passed to your security verification routine.

See also –VSAM_AUTH_EP above.

–VSAM_BUFFERSPACE

Synonyms Include: –ARCHBUFSPACE, –BUFSPACE, –BUFFERSPACE, –OUTBUFSPACE

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_BUFFERSPACE parameter defines the minimum space (in bytes) to be provided for buffers.

The IDCAMS equivalent for this command is BUFFERSPACE(size).

–VSAM_BUFFERSPACE(<buffer size>)

buffer size - Specifies the number of bytes to be provided for buffers.

Note: Access Method Services may modify the value to fit VSAM processing needs.

–VSAM_CATALOG

Synonyms Include: –ARCHCATALOG, –CATALOG, –OUTCATALOG

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_CATALOG parameter defines into which catalog the VSAM output file is to be defined. If the value is blank, then the system-defined catalog environment will be used. If the value is set to a catalog name, then the name will be used in the Define Cluster Cat(name) parameter. If the value is set to USE_ORIGINAL, then UNZIP processing will attempt to use a saved catalog attribute from the zip archive.

> **Warning**: Care must be taken when using the USE_ORIGINAL option. An inappropriate catalog may be used which does not fit within the master/user catalog structure of the target system. This may occur because the high-level-qualifier does not match the alias entries in the master catalog; either because of a change of qualifier, for example, with –UNZIPPED_DSN specifications, or because the original filename does not match the current operating environment. This can result in a file being allocated in the specified catalog, but inaccessible through normal system catalog structures. The unzip will fail and the dataset may not appear in standard catalog listings, even though the file was created.

The IDCAMS equivalent for this command is CATALOG(catname).

–VSAM_CATALOG(<catname>[/<password>])

or

–VSAM_CATALOG(USE_ORIGINAL)

catname - Specifies the name of the catalog in which the cluster is to be defined.

Password - Specifies the update or higher-level password.

USE_ORIGINAL - Specifies that UNZIP processing will attempt to use a saved catalog attribute from the Archive.

–VSAM_CISIZE

Synonyms Include: –ARCHCISZ, –ARCHCISIZE, –OUTCISZ, –OUTCISIZE, –VSAMCISZ, –VSAMCISIZE, –CISIZE

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_CISIZE parameter defines the size of the control intervals for the cluster.

The IDCAMS equivalent for this command is CONTROLINTERVALSIZE(size).

```
-VSAM_CISIZE(<size>)
```

size - Specifies (in bytes) the size of the control intervals for the cluster.

> **Note: Access Method Services may modify the value to fit VSAM processing needs.**

–VSAM_CLUSTER_TYPE

Synonyms Include: –VSAM_TYPE, –VSAMTYPE, –OUTATTR, –VSAMESDS, –VSAMKSDS, –VSAMRRDS, –ESDS, –KSDS, –RRDS

☺ - This command is also compatible with VSE. Some values may be restricted by the operating environment.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_CLUSTER_TYPE command defines the file type of a VSAM cluster.

There are three IDCAMS equivalents for this command, which include INDEXED, NONINDEXED, and NUMBERED.

```
–VSAM_CLUSTER_TYPE(ESDS|NONINDEXED|INDEXED|NUMBERED|RRDS|KSDS)
```

NONINDEXED - Entry-Sequenced VSAM file.

ESDS - Entry-Sequenced VSAM file.

INDEXED - Key-Sequenced VSAM file.

KSDS - Key-Sequenced VSAM file.

NUMBERED - Relative Record VSAM file.

RRDS - Relative Record VSAM file.

The file attributes stored in the original file will be used to create a newly extracted file unless a specification is made from the above list.

–VSAM_CODE

Synonyms Include: –OUTCODE, –OUTDATACODE

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_CODE parameter supplies a code name for the cluster or component to Access Methods Services during a DEFINE CLUSTER.

The IDCAMS equivalent for this command is CODE(code).

```
–VSAM_CODE(<name>)
```

name - The code name for the cluster or component.

–VSAM_CONTROLPW

Synonyms Include: –OUTCONTROLPW, –OUTDATACTLPW

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command specifies the control password to be passed to Access Methods Services for the definition or update of a VSAM cluster or component.

The IDCAMS equivalent for this command is CONTROLPW(password).

```
–VSAM_CONTROLPW(<pwd>)
```

pwd - This is an 8-character field specifying the control password.

–VSAM_DATA_CISIZE

Synonyms Include: –ARCHDATACISZ, –ARCHDATACISIZE, –OUTDATACISZ, –OUTDATACISIZE

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_DATA_CISIZE command provides the ability to define the size of the control intervals for the data component of a VSAM cluster.

The IDCAMS equivalent for this command is CONTROLINTERVALSIZE(size).

```
-VSAM_DATA_CISIZE(<size>)
```

size - Specifies (in bytes) the size of the control intervals for the data component.

–VSAM_DATA_EXCEPTIONEXIT

Synonyms Include: –ARCHDATAEEXT, –OUTDATAEEXT

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_DATA_EXCEPTIONEXIT parameter defines the name of your module that is given control when a problem occurs during the IDCAMS processing of the data component of the cluster.

The IDCAMS equivalent for this command is EXCEPTIONEXIT(module name).

```
-VSAM_DATA_EXCEPTIONEXIT(<exceptname>)
```

exceptname - Specifies the name of your module (phase name) that will be given control when an exception occurs.

–VSAM_DATA_FILE

Synonyms Include: –ARCHDATAFILE, –OUTDATAFILE

☺ - This command is also compatible with VSE.

- Be aware that if this command is used incorrectly, you could incur problems.

Specifies the FILE parameter of the IDCAMS DEFINE CLUSTER command used to create the data component of a new or updated ZIP archive.

The IDCAMS equivalent for this command is FILE(ddname).

–VSAM_DATA_FILE(<ddname>)

ddname - Specifies a DD statement in the JCL.

–VSAM_DATA_NAME

Synonyms Include: –ARCHDATANAME, –OUTDATANAME

☺ - This command is also compatible with VSE.

- Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_DATA_NAME command provides the ability to define a NAME parameter for the data component of a VSAM cluster.

The IDCAMS equivalent for this command is NAME(entryname).

–VSAM_DATA_NAME(<entryname>)

entryname - Specifies the name to be given to the data component of the cluster.

–VSAM_DATA_ORDERED

Synonyms Include: –ARCHDATAORD, –ARCHDATANORD, –OUTDATAORD, –OUTDATANORD

☺ - This command is also compatible with VSE.

- Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_DATA_ORDERED command provides the ability to define an ORDERED parameter for the data component of a VSAM cluster.

The IDCAMS equivalent for this command is ORDERED|UNORDERED.

–VSAM_DATA_ORDERED(<ORDERED|UNORDERED>)

ORDERED - Specifies the volumes are to be used in the order in which they were listed in the VOLUMES parameter.

UNORDERED - Specifies the volumes are not to be used in the order in which they were listed in the VOLUMES parameter.

–VSAM_DATA_PRIMARY

Synonyms Include: –ARCHDATAPRI, –OUTDATAPRI

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_DATA_PRIMARY command provides the ability to define the primary value for space allocation in the DATA component of a VSAM cluster. Note that this command is used in conjunction with –VSAM_DATA_SPACE_TYPE.

The IDCAMS equivalent for this command is CYLINDERS(primary), TRACKS(primary), or RECORDS(primary).

–VSAM_DATA_PRIMARY(<primary>)

primary - Specifies the number of units to be allocated (cylinders, tracks, records, kilobytes, or megabytes).

–VSAM_DATA_SECONDARY

Synonyms Include: –ARCHDATASEC, –OUTDATASEC

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_DATA_SECONDARY command provides the ability to define the secondary value for space allocation in the DATA component of a VSAM cluster. Note that this command is used in conjunction with –VSAM_DATA_SPACE_TYPE.

The IDCAMS equivalent for this command is CYLINDERS(secondary), TRACKS(secondary), or RECORDS(secondary).

–VSAM_DATA_SECONDARY(<secondary>)

secondary - Specifies the number of units to be allocated (cylinders, tracks, records, kilobytes, or megabytes).

–VSAM_DATA_SPACE_TYPE

Synonyms Include: –ARCHDATASPACE, –OUTDATASPACE

☺ - This command is also compatible with VSE. Some values may be restricted by the operating environment.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

For a new or updated ZIP archive, the type of allocation units may be specified using the –VSAM_DATA_SPACE_TYPE command.

Note that use of this command necessitates the use of –VSAM_DATA_PRIMARY and –VSAM_DATA_SECONDARY to define the specific extent values.

–VSAM_DATA_SPACE_TYPE(<CYL|KB|REC|MB|TRK>)

CYL - (also CYLS and CYLINDERS) allocation by cylinders.

KB - (also KILOBYTES) allocation by Kilobytes (for the ICF catalog environment only).

MB - (also MEGABYTES) allocation by Megabytes (for the ICF catalog environment only).

REC - (also RECORDS) allocation by records.

TRK - (also TRKS and TRACKS) allocation by tracks.

Also see –VSAM_DATA_PRIMARY and –VSAM_DATA_SECONDARY.

–VSAM_DATA_VOLUMES

Synonyms Include: –ARCHDATAVOL, –OUTDATAVOL, –VSAM_VOLUMES

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_DATA_VOLUMES command provides the ability to define a VOLUMES parameter for the data component of a VSAM cluster. Note that a maximum of 31 volumes are supported.

The IDCAMS equivalent for this command is VOLUMES(volser).

```
–VSAM_DATA_VOLUMES(<volser>[ <volser> …])
```

volser - Specifies a one-to-six-character volume serial number.

–VSAM_DATACLASS

Synonyms Include: N/A

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command pertains to DF/SMS allocation of new files when doing PKUNZIP processing. If you specify these classes, they will be passed to DF/SMS when dataset allocation occurs.

```
–VSAM_DATACLASS(<SMS Data Class>)
```

See IBM's DF/SMS manuals for further information about this parameter.

–VSAM_DUPLICATE_ERROR

Synonyms Include: –OUTDUPLICATES, –FAILONDUPKEYS, –IGNOREDUPKEYS

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

When extracting a file to a new VSAM keyed cluster, this command specifies the action to be taken if a duplicate key is detected.

```
–VSAM_DUPLICATE_ERROR(FAIL|IGNORE)
```

FAIL - Indicates that processing will be aborted if a duplicate key is encountered.

IGNORE - Indicates that processing will continue if a duplicate key is encountered.

–VSAM_ERASE

Synonyms Include: –ARCHERASE, –ARCHNOERASE

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_ERASE parameter defines that the data component that is being defined be erased when the cluster is deleted.

The IDCAMS equivalent for this command is ERASE|NOERASE.

–VSAM_ERASE(Y|N)

Y - YES - The IDCAMS DEFINE CLUSTER command equivalent is ERASE.

N - NO - The IDCAMS DEFINE CLUSTER command equivalent is NOERASE.

–VSAM_EXCEPTIONEXIT

Synonyms Include: –ARCHEEXT, –OUTEEXT

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_EXCEPTIONEXIT parameter defines the name of your module that is given control when a problem occurs during the IDCAMS processing of the cluster component.

The IDCAMS equivalent for this command is EXCEPTIONEXIT(module name).

-VSAM_EXCEPTIONEXIT(<entrypoint>)

exceptname - Specifies the name of your module (phase name) that will be given control when an exception occurs.

–VSAM_FILE

Synonyms Include: –ARCHFILE

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_FILE parameter defines the name of the job control DD statement that identifies the volumes that are to be used for space allocation.

The IDCAMS equivalent for this command is FILE(ddname).

-VSAM_FILE(<ddname>)

ddname - Specifies a DD statement in the JCL.

–VSAM_FOR

Synonyms Include: –ARCHFOR

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_FOR parameter defines the retention date for the cluster.

The IDCAMS equivalent for this command is FOR(days).

-VSAM_FOR(<days>)

Note that specification of either the –VSAM_TO or –VSAM_FOR commands could prevent an old ZIP archive from being deleted during an update if the old archive had an active retention period.

–VSAM_FREESPACE_CA

Synonyms Include: –ARCHFREECA, –OUTFREECA

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_FREESPACE_CA command provides the ability to define the CA-percent parameter for a key-sequenced VSAM cluster.

-VSAM_FREESPACE_CA(<ca-percent>)

ca-percent - Specifies the percentage of control area that is to be left empty.

–VSAM_FREESPACE_CI

Synonyms Include: –ARCHFREECI

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_FREESPACE_CI command provides the ability to define the CI-percent parameter for a VSAM key-sequenced cluster.

-VSAM_FREESPACE_CI(<ci-percent>)

ci-percent - Specifies the percentage of control interval that is to be left empty.

–VSAM_IMBED

Synonyms Include: –OUTIMBED, –OUTNOIMBED

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_IMBED command provides the ability to define an IMBED parameter for a VSAM cluster.

The IDCAMS equivalent for this command is IMBED|NOIMBED.

–VSAM_IMBED(Y|N)

Y - YES - Specifies that the sequence set is to be placed with the data component of a new cluster.

N - NO - Specifies that the sequence set is not to be placed with the data component of a new cluster.

–VSAM_INDEX_ATTEMPTS

Synonyms Include: –OUTINDXATT

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_ATTEMPTS parameter defines the number of password attempts that are permitted to Access Methods Services during a DEFINE CLUSTER.

The IDCAMS equivalent for this command is ATTEMPTS(number).

```
–VSAM_INDEX_ATTEMPTS(<number>)
```

number - The number of attempts that will be allowed at the console in response to a prompting message.

–VSAM_INDEX_AUTH_EP

Synonyms Include: –OUTINDXAUTH

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_AUTH_EP parameter supplies the entry point of a user security verification routine to Access Methods Services during a DEFINE CLUSTER.

The IDCAMS equivalent for this command is AUTHORIZATION(entrypoint).

```
–VSAM_INDEX_AUTH_EP(<entry point>)
```

entry point - The entry point name of your security verification routine.

–VSAM_INDEX_AUTH_STRING

Synonyms Include: –OUTINDXASTR

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_AUTH_STRING parameter supplies a string of information to be passed to your security verification routine to Access Methods Services during a DEFINE CLUSTER.

The IDCAMS equivalent for this command is AUTHORIZATION(entrypoint string).

```
–VSAM_INDEX_AUTH_STRING(<string>)
```

string - The string of information to be passed to your security verification routine.

See also –VSAM_INDEX_AUTH_EP above.

–VSAM_INDEX_CISIZE

Synonyms Include: –OUTINDXCISZ, –OUTINDXCISIZE

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_CISIZE command provides the ability to define a CONTROLINTERVALSIZE for the index component of a VSAM cluster.

The IDCAMS equivalent for this command is CONTROLINTERVALSIZE(size).

–VSAM_INDEX_CISIZE(<size>)

size - Specifies the size of the control intervals for the index component.

–VSAM_INDEX_CODE

Synonyms Include: N/A

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_CODE parameter supplies a code name for the cluster or component to Access Methods Services during a DEFINE CLUSTER.

The IDCAMS equivalent for this command is CODE(code).

–VSAM_INDEX_CODE(<name>)

name - Specifies the code name for the index component.

–VSAM_INDEX_CONTROLPW

Synonyms Include: –OUTINDXCTLPW

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command specifies the control password to be passed to Access Methods Services for the definition or update of the index component of a VSAM cluster.

The IDCAMS equivalent for this command is CONTROLPW(password).

–VSAM_INDEX_CONTROLPW(<pwd>)

pwd - Specifies a one-to-eight-character control password.

–VSAM_INDEX_EXCEPTIONEXIT

Synonyms Include: –OUTINDXEEXT

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_EXCEPTIONEXIT command provides the ability to define an EXCEPTIONEXIT parameter for the index component of a VSAM cluster.

The IDCAMS equivalent for this command is EXCEPTIONEXIT(module name).

–VSAM_INDEX_EXCEPTIONEXIT(<exceptname>)

exceptname - Specifies the name of your module (phase name) that will be given control when an exception occurs.

–VSAM_INDEX_FILE

Synonyms Include: –OUTINDXFILE

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_FILE command provides the ability to define an INDEX parameter for the index component of a VSAM cluster.

The IDCAMS equivalent for this command is FILE(ddname).

```
–VSAM_INDEX_FILE(<ddname>)
```

ddname - Specifies a DD statement in the JCL.

–VSAM_INDEX_MASTERPW

Synonyms Include: –OUTINDXMRPW

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command specifies the master password to be passed to Access Methods Services for the definition or update of the index component of a VSAM cluster.

The IDCAMS equivalent for this command is MASTERPW(password).

```
–VSAM_INDEX_MASTERPW(<pwd>)
```

pwd - This is an 8-character field specifying the master password.

–VSAM_INDEX_NAME

Synonyms Include: –OUTINDXNAME

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_NAME command provides the ability to define a NAME parameter for the index component of a VSAM cluster.

The IDCAMS equivalent for this command is NAME(entryname).

```
–VSAM_INDEX_NAME(<entryname>)
```

entryname - Specifies the name to be given to the index component of the cluster.

–VSAM_INDEX_ORDERED

Synonyms Include: –OUTINDXORD, –OUTINDXNORD

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_ORDERED command provides the ability to define an ORDERED parameter for the index component of a VSAM cluster.

The IDCAMS equivalent for this command is ORDERED|UNORDERED.

–VSAM_INDEX_ORDERED(<ORDERED|UNORDERED>)

ORDERED - Specifies the volumes are to be used in the order in which they were listed in the VOLUMES parameter.

UNORDERED - Specifies the volumes are not to be used in the order in which they were listed in the VOLUMES parameter.

–VSAM_INDEX_PRIMARY

Synonyms Include: –OUTINDXPRI

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_PRIMARY command provides the ability to define the primary value for space allocation in the INDEX component of a VSAM cluster. Note that this command is used in conjunction with –VSAM_INDEX_SPACE_TYPE.

The IDCAMS equivalent for this command is CYLINDERS(primary), TRACKS(primary), or RECORDS(primary).

–VSAM_INDEX_PRIMARY(<primary>)

primary - Specifies the number of units to be allocated (cylinders, tracks, records, kilobytes, or megabytes).

Also see –VSAM_INDEX_SECONDARY.

–VSAM_INDEX_READPW

Synonyms Include: –OUTINDXRDPW

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command specifies the read password to be passed to Access Methods Services for the definition or update of the index component of a VSAM cluster.

The IDCAMS equivalent for this command is READPW(password).

–VSAM_INDEX_READPW(<pwd>)

pwd - This is an 8-character field specifying the read password.

–VSAM_INDEX_SECONDARY

Synonyms Include: –OUTINDXSEC

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_SECONDARY command provides the ability to define the secondary value for space allocation in the INDEX component of a VSAM cluster. Note that this command is used in conjunction with –VSAM_INDEX_SPACE_TYPE.

The IDCAMS equivalent for this command is CYLINDERS(secondary), TRACKS(secondary), or RECORDS(secondary).

–VSAM_INDEX_SECONDARY(<secondary>)

secondary - Specifies the number of units to be allocated (cylinders, tracks, records, kilobytes, or megabytes).

Also see –VSAM_INDEX_PRIMARY.

–VSAM_INDEX_SPACE_TYPE

Synonyms Include: –OUTINDXSPACE

☺ - This command is also compatible with VSE. Some values may be restricted by the operating environment.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

For a new or updated ZIP archive, the type of index units may be specified using the –VSAM_INDEX_SPACE_TYPE command.

Note that use of this command necessitates the use of –VSAM_INDEX_PRIMARY and –VSAM_INDEX_SECONDARY to define the specific extent values.

The IDCAMS equivalent for this command is CYLINDERS, TRACKS, or RECORDS.

–VSAM_INDEX_SPACE_TYPE(<CYL|KB|REC|MB|TRK>)

CYL - (also CYLS and CYLINDERS) allocation by cylinders.

KB - (also KILOBYTES) allocation by Kilobytes (for the ICF catalog environment only).

MB - (also MEGABYTES) allocation by Megabytes (for the ICF catalog environment only).

REC - (also RECORDS) allocation by records.

TRK - (also TRKS and TRACKS) allocation by tracks.

Note that both the primary and secondary extents are allocated at 10 allocation units unless changed by the –VSAM_SPACE_PRIMARY or the –VSAM_SPACE_SECONDARY commands.

Also see –VSAM_INDEX_PRIMARY and –VSAM_INDEX_SECONDARY.

–VSAM_INDEX_UPDATEPW

Synonyms Include: –OUTINDXUPDPW

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command specifies the update password to be passed to Access Methods Services for the definition or update of the index component of a VSAM cluster.

The IDCAMS equivalent for this command is UPDATEPW(password).

–VSAM_INDEX_UPDATEPW(<pwd>)

pwd - This is an 8-character field specifying the update password.

–VSAM_INDEX_VOLUMES

Synonyms Include: –OUTINDXVOL

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_INDEX_VOLUMES command provides the ability to define a VOLUMES parameter for the index component of a VSAM cluster. Note that a maximum of 31 volumes are supported.

The IDCAMS equivalent for this command is VOLUMES(volser).

```
–VSAM_INDEX_VOLUMES(<volser>[ <volser> …])
```

volser - Specifies volume serial numbers sequenced by a blank.

–VSAM_KEYS

Synonyms Include: –OUTKEYS

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_KEYS command provides the ability to specify information about key fields for a VSAM key-sequenced file (ignored for entry-sequenced or relative-record files).

The IDCAMS equivalent for this command is KEYS(length offset).

```
–VSAM_KEYS(length offset)
```

length - Defines the length of a key for a key-sequenced file (255-byte maximum).

Offset - Defines the offset of the key from the front of the data record.

–VSAM_MASTERPW

Synonyms Include: –OUTMASTERPW, –OUTDATAMRPW

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command specifies the master password to be passed to Access Methods Services for the definition or update of a VSAM cluster or component.

The IDCAMS equivalent for this command is MASTERPW(password).

```
–VSAM_MASTERPW(<pwd>)
```

pwd - This is an 8-character field specifying the master password.

–VSAM_MGMTCLASS

Synonyms Include: N/A

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command pertains to DF/SMS allocation of new files when doing PKUNZIP processing. If you specify these classes, they will be passed to DF/SMS when dataset allocation occurs.

–VSAM_MGMTCLASS(<SMS Management Class>)

See IBM's DF/SMS manuals for further information about this parameter.

–VSAM_MODEL

Synonyms Include: –ARCHMODEL, –ARCHIVE_MODEL, –OUTMODEL

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command specifies that a catalog entry of a previously defined cluster is to be used as the model for a new archive.

The IDCAMS equivalent for this command is MODEL(entryname).

–VSAM_MODEL(<entryname>)

entryname - This is a 44-character entry used to specify the model.

–VSAM_ORDERED

Synonyms Include: N/A

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_ORDERED command provides the ability to define an ORDERED parameter for a VSAM cluster.

The IDCAMS equivalent for this command is ORDERED|UNORDERED.

–VSAM_ORDERED(<ORDERED|UNORDERED>)

ORDERED - Specifies the volumes are to be used in the order in which they were listed in the VOLUMES parameter.

UNORDERED - Specifies the volumes are not to be used in the order in which they were listed in the VOLUMES parameter.

–VSAM_OWNER

Synonyms Include: –ARCHDATAOWNER, –ARCHOWNER, –OUTDATAOWNER, –OUTINDXOWNER, –OUTOWNER

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_OWNER command provides the ability to define an OWNER parameter for a VSAM cluster.

The IDCAMS equivalent for this command is OWNER(owner ID).

```
-VSAM_OWNER(<owner>)
```

owner - Specifies a one-to-eight-character owner ID of the cluster.

–VSAM_READPW

Synonyms Include: –OUTREADPW, –OUTDATARDPW

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command specifies the read password to be passed to Access Methods Services for the definition or update of a VSAM cluster or component.

The IDCAMS equivalent for this command is READPW(password).

```
–VSAM_READPW(<pwd>)
```

pwd - This is an 8-character field specifying the read password.

–VSAM_RECORDSIZE

Synonyms Include: –ARCHRECORDSIZE

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_RECORDSIZE parameter defines the average and maximum lengths of the data records of a variable length file.

The IDCAMS equivalent for this command is RECORDSIZE(average maximum).

```
-VSAM_RECORDSIZE(<average> <maximum>)
```

<average> - This is the average length in bytes of each record.

<maximum> - This is the maximum length of any record.

The default for this command is (4000 4000).

It is suggested <average> = <maximum> for ***PKZIP MVS™*** processing since full-length records are written in the process. Also, a larger value for both parameters will improve ***PKZIP MVS™*** performance.

–VSAM_RECOVERY_OPT

Synonyms Include: –OUTRECOVERY, –OUTSPEED

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_RECOVERY_OPT command provides the ability to define a SPEED or RECOVERY parameter for a VSAM cluster.

The IDCAMS equivalent for this command is RECOVERY|SPEED.

–VSAM_RECOVERY_OPT(recovery|speed)

recovery - Specifies that the data component control areas are written with records that indicate an end-of-file indicator.

speed - Specifies that the data component control areas are not preformatted.

–VSAM_REPLICATE

Synonyms Include: –OUTREPLICATE, –OUTNOREPLICATE

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_REPLICATE command provides the ability to define a REPLICATE parameter for a VSAM cluster.

The IDCAMS equivalent for this command is REPLICATE|NOREPLICATE.

–VSAM_REPLICATE(Y|N)

–VSAM_REUSE

Synonyms Include: –ARCHREUSE, –ARCHNOREUSE, –ARCHDATARUS, –ARCHDATANRUS, –OUTREUSE, –OUTNOREUSE, –OUTDATARUS, –OUTDATANRUS, –OUTINDXRUS, –OUTINDXNRUS

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_REUSE parameter defines whether the newly-defined file can be opened repeatedly as a new file.

The IDCAMS equivalent for this command is REUSE|NOREUSE.

-VSAM_REUSE(Y|N)

Y - YES - Specifies that REUSE be passed to the DEFINE CLUSTER command.

N - NO - Specifies that NOREUSE be passed to the DEFINE CLUSTER command.

–VSAM_SHAREOPTIONS

Synonyms Include: –ARCHSHR, –ARCHDATASHR, –OUTSHR, –OUTDATASHR, –OUTINDXSHR, –VSAM_SHROPTS, –VSAM_SHROPT

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

–VSAM_SHAREOPTIONS(value1|value2)

The –VSAM_SHAREOPTIONS parameter defines how a file can be shared within or between systems.

The IDCAMS equivalent for this command is SHAREOPTIONS(value1 value2).

Crossregion - Specifies the level of sharing among regions.

Crosssystem - Specifies the level of sharing among systems.

–VSAM_SPACE_PRIMARY

Synonyms Include: N/A

☺ - This command is also compatible with VSE. Some values may not be restricted by the operating environment.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

For a new or updated ZIP archive, the number of allocation units in the primary extent is specified using the –VSAM_SPACE_PRIMARY command.

The default is not used if –VSAM_DATACLASS is specified.

The IDCAMS equivalent for this command is CYLINDERS(primary), TRACKS(primary), RECORDS(primary), KILOBYTES(primary), or MEGABYTES(primary).

```
–VSAM_SPACE_PRIMARY(<primary>)
```

primary - This is an 8-character field specifying the number of allocation units for the primary extent of the new or updated ZIP archive.

00000010 - Ten (cylinders) is the default.

–VSAM_SPACE_SECONDARY

Synonyms Include: N/A

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

For a new or updated ZIP archive, the number of allocation units in the secondary extent is specified using the –VSAM_SPACE_SECONDARY command. If specified, the data unit number must not be 0.

The default is not used if –VSAM_DATACLASS is specified.

The IDCAMS equivalent for this command is CYLINDERS(secondary), TRACKS(secondary), RECORDS(secondary), KILOBYTES(secondary), or MEGABYTES(secondary).

```
–VSAM_SPACE_SECONDARY(<secondary>)
```

secondary - This is an 8-character field specifying the number of allocation units for the secondary extent of the new or updated ZIP archive.

00000010 **- Ten (cylinders) is the default.–VSAM_SPACE_TYPE**

Synonyms Include: N/A

☺ - This command is also compatible with VSE. Some values may not be restricted by the operating environment.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

For a new or updated ZIP archive, the type of allocation units may be specified using the –VSAM_SPACE_TYPE command. Note the default is not used when –VSAM_DATACLASS is specified.

The IDCAMS equivalent for this command is CYLINDERS, TRACKS, or RECORDS.

–VSAM_SPACE_TYPE(<CYL|KB|REC|MB|TRK>)

CYL - (also CYLS and CYLINDERS) allocation by cylinders.

KB - (also KILOBYTES) allocation by Kilobytes (for the ICF catalog environment only).

MB - (also MEGABYTES) allocation by Megabytes (for the ICF catalog environment only).

REC - (also RECORDS) allocation by records.

TRK - (also TRKS and TRACKS) allocation by tracks.

This command specification can be overridden at the data level by the –VSAM_DATA_SPACE_TYPE command. At the data level, the corresponding cluster information is not recognized.

–VSAM_SPANNED

Synonyms Include: –ARCHSPANNED, –ARCHNONSPANNED

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_SPANNED parameter defines whether the maximum length of a data record can be greater than the control interval size.

The IDCAMS equivalent for this command is SPANNED|NONSPANNED.

–VSAM_SPANNED(Y|N)

Y - YES - The IDCAMS DEFINE CLUSTER command equivalent is SPANNED.

N - NO - The IDCAMS DEFINE CLUSTER command equivalent is NONSPANNED.

–VSAM_STORCLASS

Synonyms Include: N/A

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command pertains to DF/SMS allocation of new files when doing PKUNZIP processing. If you specify these classes, they will be passed to DF/SMS when dataset allocation occurs.

–VSAM_STORCLASS(<SMS Storage Class>)

See IBM's DF/SMS manuals for further information about this parameter.

–VSAM_TO

Synonyms Include: –ARCHFOR, –ARCHTO, –OUTFOR, –OUTTO

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_TO parameter defines the retention date for the cluster.

The IDCAMS equivalent for this command is TO(date).

–VSAM_TO(<date>)

date - Specifies the date until which the cluster is to be retained.

> **Note: The specification of either the –VSAM_TO or –VSAM_FOR commands could prevent an old ZIP archive from being deleted during an update if the old archive had an active retention period.**

–VSAM_UPDATEPW

Synonyms Include: –OUTUPDATEPW, –OUTDATAUPDPW

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

This command specifies the update password to be passed to Access Methods Services for the definition or update of a VSAM cluster or component.

The IDCAMS equivalent for this command is UPDATEPW(password).

```
–VSAM_UPDATEPW(<pwd>)
```

pwd - This is an 8-character field specifying the update password.

–VSAM_WRITECHECK

Synonyms Include: –ARCHWRITECHK, –ARCHNOWRITECHK, –ARCHDATAWCK, –ARCHDATANWCK, –OUTDATAWCK, –OUTDATANWCK, –OUTWRITECHK, –OUTNOWRITECHK

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –VSAM_WRITECHECK parameter defines whether to verify the transfer of records written to the cluster.

The IDCAMS equivalent for this command is WRITECHECK|NOWRITECHECK.

```
–VSAM_WRITECHECK(WRITECHECK|NOWRITECHECK)
```

WRITECHECK - The IDCAMS DEFINE CLUSTER command equivalent is WRITECHECK.

NOWRITECHECK - The IDCAMS DEFINE CLUSTER command equivalent is NOWRITECHECK.

–ZIPPED_DSN

Synonyms Include: –NIA

☺ - This command is also compatible with VSE.

☠ - Be aware that if this command is used incorrectly, you could incur problems.

The –ZIPPED_DSN command specifies one or more MVS file names and how they are to be renamed for the associated ZIP file. More than one file may be referenced in one command by the use of wildcard characters. The default depends on the MVS file type and other situations as outlined below.

```
–ZIPPED_DSN(<MVS name>,<Archive name>)
```

MVS name - This is one entry representing one or more MVS file names. The maximum character length is 54 characters. Spaces are not valid. Wild characters ("*") may be used here for two purposes. One, the character is used to identify more than one file. <MVS name> = MYFILE.NEW* will represent MYFILE.NEW1, MYFILE.NEW2, MYFILE.NEW3, and so on. The second purpose is to identify what part of the <MVS name> is to be used in the <Archive name>. A matching wild character in the <Archive name> indicates the corresponding part of the <MVS name> is duplicated in the <Archive name>. See the table below for examples.

Archive name - This is the format for the associated ZIP file name(s). The maximum character length is 80 characters. Imbedded spaces are supported. The entry contains the ZIP file component name and may contain wild characters("*") and ignore characters ("+"). Each wild character matches a wild character in the <MVS name> and copies those characters from the <MVS name> onto the ZIP archive name at the "*" location. Each ignore character matches with a wild character in the <MVS name> and does not copy those characters from the <MVS name> into the Zip archive name at the "+" location. (See below).

See the table below for –ZIPPED_DSN examples:

<MVS file>	<MVS name>,<Archive name>	ZIP File name results
MVS.SEQ.INFO	MVS.SEQ.INFO,ZIP/INF	ZIP/INF
MVS.SEQ.INFO	MVS.SEQ.INFO,ZIP.EXT	ZIP.EXT
MVS.PDS(MEM1)	MVS.PDS(*),ZIP/LIB/*	ZIP/LIB/MEM1
MVS.PDS(MEM2)	*Allow to default*	ZIP/LIB/MEM2
MVS.PDS(MEMN)	*.*(MEMN),*/*.DAT	MVS/PDS.DAT
MVS.PDS(MEMN)	*.*(*),*/*/*.DAT	MVS/PDS/MEMN.DAT
MVS.PDS(MEMN)	*.PDS(*),*/*/INFO	MVS/MEMN/INFO
MVS.PDS(MEMN)	*(*),+*.INF	MEMN.INF
MVS.SEQ.INFO	*.*.INFO,*.*	MVS.SEQ
MVS.SEQ.INFO	*.*.DATA,+*.INF	SEQ.INF

More than one –ZIPPED_DSN command may be used in one execution to match various input/output combinations. File names are converted based on the order of occurrence of –ZIPPED_DSN commands. In the following example, the file MYFILE.INPUT.DAT would be processed by the second –ZIPPED_DSN command.

```
MYFILE.INPUT.DAT
-ZIPPED_DSN(*.IN.*,*/*)
-ZIPPED_DSN(*.INPUT,*/*.DAT) (here the file is processed)
-ZIPPED_DSN(*.DATA,*/DAT)

ZPAM253I ADDED     File MYFILE.INPUT.DAT
ZPAM254I   as MYFILE/INPUT/DAT.TXT
ZPAM255I   (DEFLATED 62%/61%)
```

This would create the ZIP file: MYFILE/INPUT.DAT.

Notes for –ZIPPED_DSN

File names are converted based on the order of occurrence of –ZIPPED_DSN commands. For example, the file MYFILE.INPUT.DAT would be processed by the second –ZIPPED_DSN command in the following example.

–ZIPPED_DSN(*.IN.*,*/*)

–ZIPPED_DSN(*.INPUT,*/*.DAT) (here the file is processed)

–ZIPPED_DSN(*.DATA,*/DAT)

This would create the ZIP file: MYFILE/INPUT.DAT.

Care must be taken when coding this command to achieve a desired result. Examples of errant coding techniques follows:

Example:	
Given the PDS member:	MVS.PDS(MEMBER)
and a command of:	-ZIPPED_DSN(*,*.TXT)
the ZIP archive will be:	MVS.PDS(MEMBER).TXT (an invalid filename)

or

Given the PDS member:	MVS.PDS(MEMBER)
and a command of:	-ZIPPED_DSN(MV*,PD*.TXT)
the ZIP archive will be:	PDS.PDS(MEMBER).TXT (an invalid filename)

When coding this command for new filename translation, the –SIMULATE command can be used in test runs to ensure that the desired results are being achieved without the processing time associated with compression and archiving.

The allowable number of –ZIPPED_DSN commands is determined by each command's storage requirements of approximately 256 bytes.

The allowable number of wild characters ("*") is determined by the <MVS name> format. Extra wild characters adjacent to other wild characters are not supported. The maximum number of wild characters in the <MVS name> is 28.

There must be a match of wild characters in the <Archive name> to the <MVS name> or unpredictable results may occur. Any extra wild characters in the <MVS name> are ignored. For example, a null filename may result from –ZIPPED_DSN(*,+), which instructs all MVS DSN characters to be deleted.

Defaults for –ZIPPED_DSN

If the –ZIPPED_DSN command is not specified, the default ZIP file name will depend on the MVS file type.

NonVSAM files

- Periods for all dataset types and the left-parenthesis associated with PDS and PDSE member formats will be converted to the active –ZIPPED_DSN_SEPARATOR character. The right-parenthesis for member name designations will be ignored. For example:

```
-ARCHIVE(MY.TEMP.ZIP)
-ACTION(UPDATE)
DEV.IVP.SEQ
DEV.PROJ.SRC(ASCIIUS)
ZPAM030I OUTPUT Archive opened:  MY.TEMP.ZIP
ZPAM253I ADDED File DEV.IVP.SEQ
ZPAM254I as DEV/IVP/SEQ
```

```
ZPAM255I (DEFLATED 78%/78%)
ZPAM253I ADDED File DEV.PROJ.SRC(ASCIIUS)
ZPAM254I as DEV/PROJ/SRC/ASCIIUS
ZPAM255I (DEFLATED 62%/61%)
```

Note that the command –PATH(N) may change the expected path name and –ZIPPED_DSN_SEPARATOR can create a file with the command's specified separators. If the –ZIPPED_DSN command is not specified, verify that the path and separators are specified correctly.

VSAM Clusters for –ZIPPED_DSN

The –ZIPPED_DSN specifications are also applied to the index and the data levels of file names, which are stored as attributes within the archive. Separate –ZIPPED_DSN commands may not be applied to the component names.

The created DATA and INDEX names are appended with ".DATA" and ".INDX" respectively. The MVS separator "." is used rather than the active value of –ZIPPED_DSN_SEPARATOR.

```
-ACTION(ADD)
-ARCHIVE_DSN(MAS.TEMP.ZIP)
-ARCHIVE_DSORG(PS)
-ACTION(UPDATE)
MAS.TEST.KSDS
-ZIPPED_DSN(MAS.*.KSDS,MAS/NEWLVL/KSDS)
ZPAM030I OUTPUT Archive opened:  MAS.TEMP.ZIP
ZPAM253I ADDED     File MAS.TEST.KSDS
ZPAM254I   as MAS/NEWLVL/KSDS
ZPAM255I   (DEFLATED 91%/91%)

Resulting values:

-ACTION(VIEWDETAIL)
ZPAM001I Filename: MAS/NEWLVL/KSDS
ZPAM332I VSAM Data    Name: MAS.NEWLVL.KSDS.DATA
ZPAM333I VSAM Index   Name: MAS.NEWLVL.KSDS.INDEX
```

–ZIPPED_DSN_SEPARATOR

Synonyms Include: –NIASEP

To specify the separator to be used in the created ZIP archive name, –ZIPPED_DSN_SEPARATOR command is used. The default is "/" or Hex '2F' to conform to ***PKZIP***® Specifications, which provides for cross-platform compatibility. This creates a file name where each MVS qualifier is converted to a directory name. For example, period separators are changed to the specified separator character.

–ZIPPED_DSN_SEPARATOR(<sepchar>)

sepchar - This is the character to be used as a separator between components in the ZIP file name. It may be coded in one of two formats:

EBCDIC Display Character - Where the character is a single EBCDIC character. This will be translated with the –TRANSLATE_TABLE_FILEINFO table to ASCII before used in the ZIP file. A "/" is the default character.

X'Hex' - Where the actual ASCII character is specified in hex and is not translated before placed in the ZIP file. A hex character of "2F" is the default character.

Note: Use of a separator character other than the default should be done with consideration of the targeted PKUNZIP system. Unexpected results may occur during an extract if the filename does not adhere to the target system's file naming standards.

Example:	
Given the PDS member:	XXX.YYY(ZZZ)
and a command of:	(**not** specified: using the default value of "/")
the ZIP archive will be:	XXX/YYY/ZZZ

Example:	
Given the PDS member:	XXX.YYY(ZZZ)
and a command of:	-ZIPPED_DSN_SEPARATOR($)
the ZIP archive will be:	XXXYYYZZZ

Chapter 10. - Processing with GZIP

What is GZIP?

GNU Zip is a different standard for handling compressed file data in an Archive. Support for the GZIP standard can be found in various utilities for many platforms. This format is not compatible with ***PKZIP for MVS™*** archives; however, ***PKZIP for MVS™*** provides limited support for GZIP archives (Information regarding RFC processes for information interchange with regard to GZIP can be found at www.faqs.org/rfcs).

RFC 1951 is the specification that describes DEFLATE compressed data format that is to be used with GZIP archives. ***PKZIP MVS™*** creates a compression stream that is compatible with this format.

RFC 1952 describes the GZIP Archive format specifications. Differences from ***PKZIP MVS™*** Archives include:

- All GZIP filenames must be represented in lower case.
- Both Binary and Text data are supported by GZIP, however, the LATIN-1 translation table is the defined standard for EBCDIC/ASCII filename translation (ISO 8859-1).

Why use GZIP?

GZIP may be useful when file exchanges to a platform only having a GZIP support utility is necessary. In addition, the ***PKZIP for MVS™*** archive format is restricted to a 4-gigabyte file size limit, whereas GZIP is not. (The PKZIP Archive format has 4-byte binary size fields which limits byte-counts to 4*1024*1024 characters. The total archive size is limited to this total size, which means the total data capacity is less than that total because of PKZIP Directory information. GZIP trailer information uses 8-byte binary length fields which dramatically increases its capacity.)

Although GZIP has an almost limitless capacity, it has other significant limitations that make it less attractive than ***PKZIP for MVS™*** for most applications.

GZIP lacks a “directory” of the files contained within it. In addition, files (GZIP nomenclature is different that that of PKZIP. Whereas “files” are stored in a PKZIP archive, these data entities are (known as “members” within a GZIP archive) contained within a GZIP Archive can only be found in a serial fashion.

The file information controls provided in GZIP archives cannot be fully reported on until the entire data stream is decompressed.

This format may not be recognized by other products providing ***PKZIP®*** archive support, and thereby restricts its cross-platform usefulness.

GZIP should be used when compressing a single, large dataset, such as, a file likely to exceed the 4Gb limit, or 2Gb where the target system’s implementation of ***PKZIP®*** does not support files in excess of 2Gb.

PKZIP MVS™ Implementation Notes for GZIP

The DEFLATE compression algorithm used in GZIP is similar to the compression logic used in ***PKZIP for MVS***™ archives. The archive format is compatible with GZIP processes running on other platforms, although extensions provided by ***PKZIP for MVS***™ may not be supported by other utilities.

The standard GZIP archive format maintains a header entry at the beginning that describes the name of the file and a timestamp. A CRC integrity value is also maintained, however, this value is stored at the end of the file along with the original size of the input file.

GZIP Restrictions

The ***PKZIP MVS***™ implementation for GZIP is restricted to 1 file within an Archive. For this reason, only the –ADD Action for a new archive is supported. Attempting to FRESHEN an existing file within an archive, adding additional files, or deleting a file from an archive should not be attempted.

Only the first file in a GZIP archive from another platform will be processed by UNZIP processing. For this reason, when creating GZIP archives on other platforms with MVS as the target system, only place one file in each GZIP archive file.

An existing archive must be processed in accordance with its archive type, such as, ***PKZIP MVS***™ or GZIP. For example, an existing ***PKZIP MVS***™ archive cannot have GZIP data appended to it. A message will be issued and processing will be terminated if this rule is not followed.

VIEW processing will not report the CRC or file size information because of the way GZIP archives hold the information.

–COMPRESSION_LEVEL(STORE) is not part of the GZIP standard, and is therefore ignored by the compression engine.

GZIP Extensions

As a proprietary extension, standard (96 bit) password encryption support is provided beyond the RFC standard.

File attributes can be stored in the GZIP archive (just as they are in a ***PKZIP MVS***™ archive), so that the file can be reconstructed during EXTRACT processing.

Filename control commands may be used, for example, –ZIPPED_DSN may be used, although lower-case translation of the resulting name will be done to conform to GZIP requirements.

During EXTRACT processing, if the GZIP archive does not contain a file name (not required by GZIP specifications), then a filename will be constructed with a low-level qualifier (or PDS/PDSE member name) of “GZOUT”, by using the input archive name as the base. This pseudo-name will then be processed by filename-modifying commands such as –UNZIPPED_DSN. (See also –GZIP_SUFFIX in the Commands chapter).

Although the default specification for GZIP processing is to handle data as BINARY, ***PKZIP MVS***™ will use the –DATA_TYPE command with DETECT or TEXT processing.

Although –MULTI_THREAD_LIMIT is ignored for GZIP processing (because only 1 file can be compressed), multi-tasking is still performed for input file reads, data compression and archive file writes to maximize processing throughput.

Although the GZIP standard does not support directory levels in the filename, many products (including ***PKZIP MVS™***) support this as an extension.

Although the format of the timestamp in the archive is in UNIX-format, and is by specification to be UTC, ***PKZIP MVS™*** will honor the –TIMESTAMP command.

Processing GZIP Archives

In general, a GZIP archive must be processed only in GZIP mode and with only one GZIP "member." When creating a GZIP Archive, specify –GZIP in the command stream (or use a defaults module with the GZIP value set). UNZIP processing in ***PKZIP for MVS™*** will automatically detect the GZIP header and process it accordingly.

If zipping a file for transport to another platform that does not support the extensions provided by ***PKZIP for MVS™***, then commands should be used to nullify those extensions, for example, –NOATTRIB, –NOPATH.

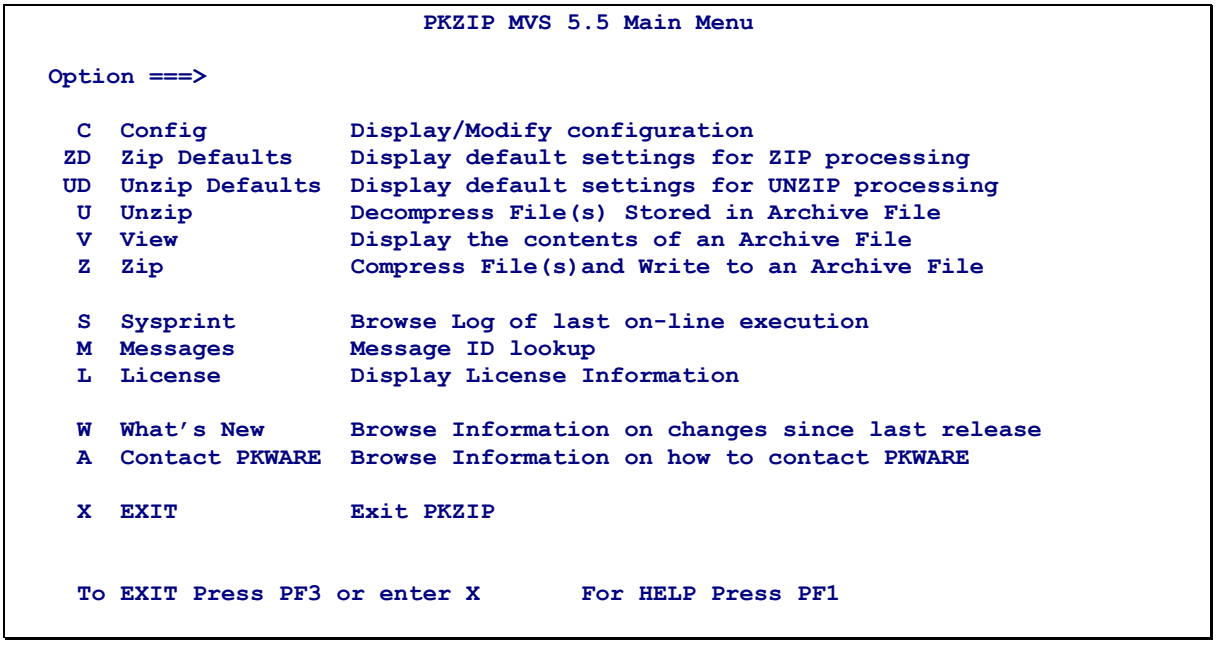
Chapter 11. - Using the ISPF Interface

Getting Started with the ISPF Interface

When the ***PKZIP for MVS™*** ISPF interface is started for the first time the Configuration Menu will be displayed. On subsequent use, the first display is the Main Menu. An example of this menu is shown below. From this panel the desired function can be selected by entering the letter associated with that function.

To display the help information, press PF1 from any panel in the ISPF interface and the help panel for that function will be displayed.

To end the ***PKZIP for MVS™*** ISPF session, press PF3 or enter "X" while the main menu is displayed.


PKZIP MVS 5.5 Main Menu
Option ===>
C Config Display/Modify configuration
ZD Zip Defaults Display default settings for ZIP processing
UD Unzip Defaults Display default settings for UNZIP processing
U Unzip Decompress File(s) Stored in Archive File
V View Display the contents of an Archive File
Z Zip Compress File(s)and Write to an Archive File
S Sysprint Browse Log of last on-line execution
M Messages Message ID lookup
L License Display License Information
W What's New Browse Information on changes since last release
A Contact PKWARE Browse Information on how to contact PKWARE
X EXIT Exit PKZIP
To EXIT Press PF3 or enter X For HELP Press PF1

Configuration (Option 'C')

The ***PKZIP for MVS™*** ISPF interface requires configuration information to function correctly in the user environment. Upon initial use of the ISPF interface, the Configuration Menu will be displayed regardless of the option selected. The following is an example of the Menu.

```
                          PKZIP for MVS 5.5 Configuration
OPTION  ===>
 Specify load library..: 'PKZIP.MVS.LOAD'

 Specify defaults file(s):               Specify defaults module: ACZDFLT
   ZIP processing......: 'PKZIP.MVS.INSTLIB(CMDZIP)'
   UNZIP processing....: 'PKZIP.MVS.INSTLIB(CMDUNZIP)'

 Miscellaneous:                          Sysprint allocation info:
   Use TSO Prefix (Y/N): Y                  Pri : 3
   Lowest Acceptable RC: 4   (0,4,8)        Sec : 1
                                            Type: CYLS    (BLKS,TRKS,CYL)

 Job Card information:
//JOBNAME JOB 'ACCOUNTING INFO',CLASS=A,REGION=8M,
//        MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID

   To EXIT Press PF3       For HELP Press PF1
```

The configuration panel is shown above. There are several configuration data fields on this panel. They are:

Load Library -The library that contains the executable code for ***PKZIP MVS™***. The default is the installed load library.

Defaults Module - The module listed here will be used as the installed defaults for all PKZIP or PKUNZIP jobs generated by ISPF. The default module is ACZDFLT.

Defaults Files - The files that contain any overrides to the installed defaults. There is one for ZIP processing and one for UNZIP processing. The default file names are *dsnhlq*.INSTLIB(CMDZIP) and *dsnhlq*.INSTLIB(CMDUNZIP) (where *dsnhlq* is the high level qualifier specified during installation).

TSO Prefix - This field controls the use of the TSO prefix. Specify 'Y' to have the value of the TSO prefix appended to all unquoted dataset names as the high level qualifier. If NOPREFIX is specified in the TSO PROFILE, then the value of this field is ignored.

Lowest Acceptable RC - This field controls the display of the generated output of a foreground execution. If the return code of the execution is greater than the number entered in this field, then the output is automatically displayed after the run.

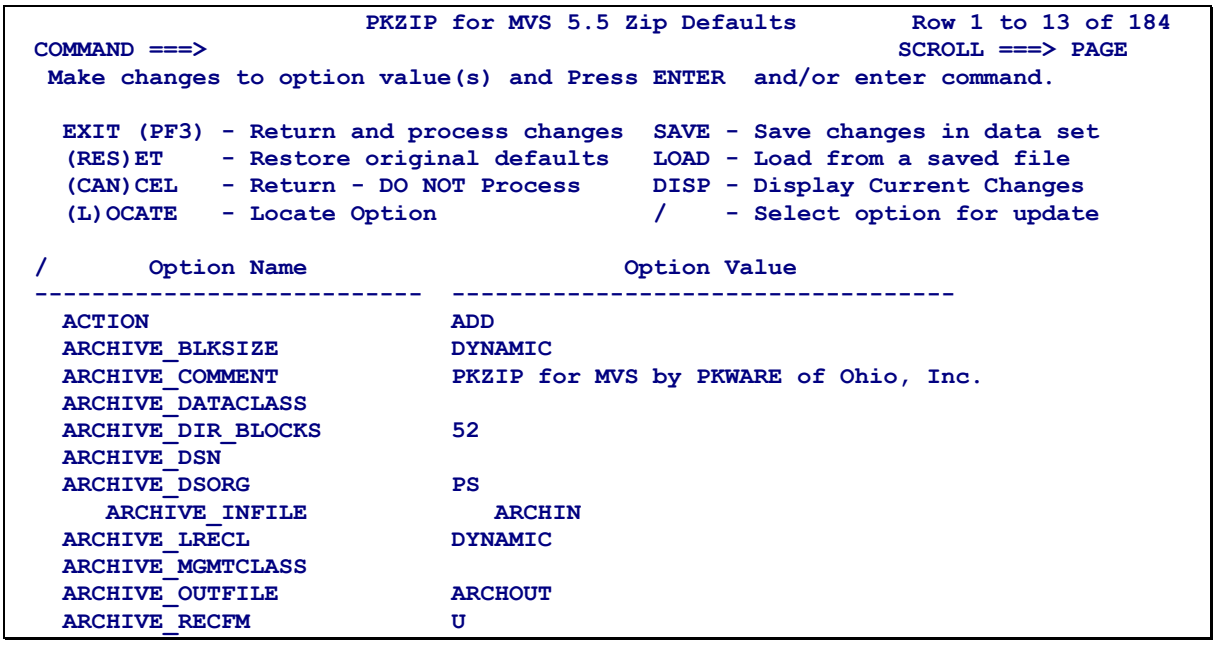
Sysprint Allocation - This information is used to set the default size for the SYSPRINT (output) file.

Job Card - This field contains the default job card that will be used in all batch jobs generated by the ***PKZIP MVS™*** ISPF interface.

Defaults (Options ZD and UD)

As explained previously PKZIP MVS defaults are provided at installation time. These are reflected in the table displayed by options ZD and UD. Option ZD will display the defaults in place for ZIP processing and option UD will display the UNZIP defaults.

When either option ZD or UD is selected, the defaults are displayed in a scrollable table. An example is shown below. The defaults can be changed and will override the installed defaults for the remainder of this ISPF ***PKZIP for MVS™*** session or until a LOAD or RESET command is entered. Use the CANCEL command to return to the calling function without processing the changes. All changes made prior to the cancel will remain in effect until RESET or until the ISPF session is terminated.



```
                      PKZIP for MVS 5.5 Zip Defaults        Row 1 to 13 of 184
COMMAND ===>                                                SCROLL ===> PAGE
 Make changes to option value(s) and Press ENTER  and/or enter command.

  EXIT (PF3) - Return and process changes  SAVE - Save changes in data set
  (RES)ET    - Restore original defaults   LOAD - Load from a saved file
  (CAN)CEL   - Return - DO NOT Process     DISP - Display Current Changes
  (L)OCATE   - Locate Option               /    - Select option for update

/       Option Name                    Option Value
--------------------------  ----------------------------------
  ACTION                    ADD
  ARCHIVE_BLKSIZE           DYNAMIC
  ARCHIVE_COMMENT           PKZIP for MVS by PKWARE of Ohio, Inc.
  ARCHIVE_DATACLASS
  ARCHIVE_DIR_BLOCKS        52
  ARCHIVE_DSN
  ARCHIVE_DSORG             PS
     ARCHIVE_INFILE            ARCHIN
  ARCHIVE_LRECL             DYNAMIC
  ARCHIVE_MGMTCLASS
  ARCHIVE_OUTFILE           ARCHOUT
  ARCHIVE_RECFM             U
```

Use PF7 and PF8 or the UP and DOWN commands to scroll the table display.

Only the first 34 characters of an option value is displayed. If the option value exceeds 34 characters, then a '+' will be displayed at the end of those 34 characters indicating that the option value is longer than the display field. The entire length of the field is maintained when the changes are processed. Only the display is truncated.

Primary Commands

Below are the commands that can be entered on the defaults panel.

CANCEL - This command allows you to return from the default options display. without generating and All changes made prior to the CANCEL command will remain. To change them back to the original defaults use the RESET command as explained above.

DISP - This command gathers all the changes to the default options and format a scrollable display showing the options, their current value and the origin of the change. The origin can be DS, loaded from a dataset, CD, changed on the defaults panel, or AV, changed by the Advanced Option feature on the ZIP and UNZIP options which are explained later in this chapter. An example of this display is shown below.

EXIT - End the defaults display and return to the ***PKZIP MVS™*** main menu. Pressing PF3 has the same results.

LOAD - This command loads default settings that were previously saved in a dataset using the SAVE command. You are prompted to enter the dataset name and member name. First a RESET is done to clear any previously changed defaults and then the default option values saved in the dataset entered are loaded and the displayed table is updated. Any options changed by the LOAD are flagged with the string '**Loaded**'. These defaults will remain in effect until this ***PKZIP MVS™*** session is ended or the RESET command is entered.

LOCATE - This command positions the table display to a particular default option or to a default option beginning with a certain string. For example, by entering LOCATE C the table display will be positioned so that the first default option beginning with the letter 'C' will be the first line displayed. This command can be truncated to LOC or L.

RESET - This command resets any changes made using this option and restore the defaults as they were installed and/or modified by the systems programmer. This command can be truncated to RES.

SAVE - This command prompts you for a dataset name and member. Then any changes made to the defaults subsequent to the SAVE command are written to the dataset entered. That dataset can then be reloaded using the LOAD command explained below.

```
                    Display of Changed Zip Defaults          Row 1 to 4 of 4
COMMAND ===>                                                SCROLL ===> PAGE

  The following options have been changed from the original defaults.

  Source of changes:   DS - Loaded from data set
                       CD - Changed default panel
                       AV - Zip or UnZip Advanced options

/       Option Name               Current Option Value            Origin
--------------------------- ---------------------------------- -------------
MEMORY_MODEL                LARGE                               AV
ACTION                      FRESHEN                             DS
LOGGING_LEVEL               VERBOSE                             DS
ARCHIVE_DSORG               PE                                  CD
```

Changing Default Options

The default option values vary depending on the option being changed. There are seven types of options: A Y/N option, a numeric option, an EBCDIC character, a dataset name, clear text, a list of volumes, and an option list. Each of these are explained below. An option of any type must be selected for update by first typing a '/' in the field at the beginning of line where the desired option is displayed and pressing ENTER. After the option is selected and it's type is determined, then the update proceeds as explained below.

Y/N - The value of the selected YES/NO option is toggled. If it is currently a 'Y', then it is made a 'N' and vice versa.

Numeric - When an option with a numeric value is selected, then a 'pop-up' panel is displayed where the desired numerical value can be entered.

EBCDIC - When an option value is a single EBCDIC character, a 'pop-up' panel is displayed where the desired character can be entered.

Dataset Name - If the value of an option is a dataset name, then a 'pop-up' panel is displayed allowing the name of a dataset to be entered. The dataset name can be in the form of MY.DATA.SET.NAME or MY.DATA.SET.NAME(MEMBER).

Text - If an option value is character or text information, then a 'pop-up' panel is displayed allowing the desired text to be entered. Text can be up to 255 characters depending on the option.

Volume List - Some option values are lists of volume serial numbers. Selecting an option of this type will cause a 'pop-up' panel to display where from 1 to 31 volume serial numbers can be entered.

Option List - Several options have a list of valid values. When an option of this type is selected for update, a scrollable panel is displayed showing all of the valid values for that option. The desired value can then be selected by placing a '/' beside the desired value.

Changes entered for the updates above will be identified on the panel by the string '**Changed**.

Including Changed Defaults

Any ZIP and/or UNZIP default options changed using this ***PKZIP for MVS™*** option are included in every corresponding ZIP and/or UNZIP foreground and batch job generated during this ***PKZIP for MVS™*** ISPF session. A ***PKZIP for MVS™*** ISPF session is defined from the time the main menu is displayed until it is exited. The proper commands are generated and included in the appropriate input stream.

View Archive (Option 'V')

This option is used to view information about the files contained in a zip archive. The information is formatted in a scrollable table and displayed on a panel. The table can be scrolled 'UP', 'DOWN', 'LEFT', and 'RIGHT' using the commands (or PF7, PF8, PF10, and PF11). The information displayed about each archived file spans 3 panels and scrolling LEFT or RIGHT will successively display the three panels and the associated archived file information. The initial panel for option 'V' is shown below. There are also several line commands that can be used to browse, view, extract, display file information, or delete the selected file.

```
                          PKZIP for MVS 5.5 View Archive
Command ===>

 Enter name of archive to be viewed:
     Archive Name . .: 'ARCHIVE.FILE'
     Dataset Filter .:

 Enter VIEW Options:
     View Type . .: V ( V - View, D - Detail, B - Brief)
     Sort Output  : N ( Y - Yes, N - No)
     Sort Field . :   ( D - Date, N - Name, O - Offset, P - Percent, S - Size)
     Sort Order . :   ( A - Ascending, D - Descending)

 Processing Mode. : F ( F - Foreground, B - Batch)
```

To EXIT Press PF3 or enter X For HELP Press PF1

Setting VIEW Options

The panel shown above is used to specify options for the VIEW operation. The individual fields are as follows:

Archive Name - Enter the name of the archive to be viewed. It can be in the form of DATA.SET.NAME or DATA.SET.NAME(MBR). Standard dataset naming conventions apply. Place the dataset name in single quotes ('…') to prevent using the TSO prefix as the first qualifier. This option can be turned off using the Configuration option explained earlier.

Dataset Filter - This field is used to specify a wildcard type filter used to limit the number of datasets displayed. If this field is entered, only those datasets matching the filter will be displayed on the VIEW information panels.

View Type - If a 'V' is selected, then the files within the selected archive are displayed on the scrollable panels shown within this chapter. The information displayed on the panels is obtained from a –ACTION(VIEWDETAIL) command. If a 'B' is selected then a –ACTION(VIEWBRIEF) command is executed and the output print file is displayed using ISPF browse. The 'D' option generates the same command as the 'V' option, but the output is browsed instead of being displayed in an ISPF table.

Sort Output - The displayed file list can be sorted prior to being displayed by entering a 'Y' in this field. When this field is a 'Y' then the following two fields are used to specify sort options.

Sort Field - This field is used to specify which display fields to sort on. Enter a 'D', 'N', 'O', 'P', or 'S' to sort on date, file name, offset, percent compressed, and compressed size, respectively.

Sort Order - Specify 'A' for ascending or 'D' for descending.

Processing Mode - Specifying a 'F' in this field will run the view job as a foreground task. Specifying a 'B' will build JCL for a batch job. The JCL will be displayed so it can reviewed and/or modified before submission. Only the foreground task will display the output on the panels.

To exit the VIEW operation and return to the main menu, press PF3. Help for the VIEW function can be obtained by pressing PF1. Pressing PF3 on any of the information display panels will return to the main VIEW panel.

```
                        PKZIP for MVS 5.5 View Archive        Row 1 to 7 of 48
Command ===>                                                 SCROLL ===> CSR
   Name of Archive : 'PKZIP.TEST.ZIP'

   Primary commands: LOCATE to position list or  SORT to sort list.
   Enter line command or '/' for list of valid line commands.
   Press PF1 for HELP.

Cmd  File Name     Zipped        Zipped  Unzipped  Comp   Type Volume(s) Message
                  Date/Time       Size     Size    Ratio
               ----------------  ------   ------   -----  ---- -------
 _   PKZIP/TEST/PDS/DELCSI
               1/24/2001  10:42  456      3281      86%   TEXT TSO002
 _ + PKZIP/TEST/PDS/DELLINK
               1/24/2001  10:42  8010     85855     90%   TEXT TSO002
 _   PKZIP/TEST/PDS/DELNUC
               1/24/2001  10:42  8010     85855     90%   TEXT TSO002
 _ + PKZIP/TEST/PDS/DELNVSM
               1/24/2001  10:42  365      1477      75%   TEXT TSO002+
 _   PKZIP/TEST/PDS/DELUCAT
               1/24/2001  10:42  314      1067      70%   TEXT TSO002+
 _   PKZIP/TEST/PDS/DELVSAM
               1/24/2001  10:42  278      1067      73%   TEXT TSO002
 _   PKZIP/TEST/PDS/DIAGBCS
               1/24/2001  10:42  230      739       68%   TEXT TSO002
```

```
                        PKZIP for MVS 5.5 View Archive        Row 1 to 7 of 48
Command ===>                                                 SCROLL ===> CSR
   Name of Archive : 'PKZIP.TEST.ZIP'

   Primary commands: LOCATE to position list or  SORT to sort list.
   Enter line command or '/' for list of valid line commands.
   Press PF1 for HELP.

Cmd  File Name  Ds    Rec Record Block          Space          Date       Last
                Org   Fmt  Size  Size    Prim Sec Dir Unit  Created    Referenced
                ----  --- ------ ------  ---- --- --- ----  ---------- ----------
 _   PKZIP/TEST/PDS/DELCSI
                PO    FB  80     27920   5    2   200 CYL   2001/01/24 2001/01/24
 _   PKZIP/TEST/PDS/DELLINK
                PO    FB  80     27920   5    2   200 CYL   2001/01/24 2001/01/24
 _   PKZIP/TEST/PDS/DELNUC
                PO    FB  80     27920   5    2   200 CYL   2001/01/24 2001/01/24
 _   PKZIP/TEST/PDS/DELNVSM
                PO    FB  80     27920   5    2   200 CYL   2001/01/24 2001/01/24
 _   PKZIP/TEST/PDS/DELUCAT
                PO    FB  80     27920   5    2   200 CYL   2001/01/24 2001/01/24
 _   PKZIP/TEST/PDS/DELVSAM
```

```
                PO   FB  80      27920  5    2   200 CYL  2001/01/24 2001/01/24
  _   PKZIP/TEST/PDS/DIAGBCS
                PO   FB  80      27920  5    2   200 CYL  2001/01/24 2001/01/24
```

```
                          PKZIP for MVS 5.5 View Archive        Row 1 to 7 of 48
Command ===>                                                     SCROLL ===> CSR
  Name of Archive : 'PKZIP.TEST.ZIP'

  Primary commands: LOCATE to position list or  SORT to sort list.
  Enter line command or '/' for list of valid line commands.
  Press PF1 for HELP.

 Cmd  File Name  Compression   CRC     Compressed          Needed to
                   Method              By                  Extract
                 ----------- -------- ------------------  --------------------
  _   PKZIP/TEST/PDS/DELCSI
                 DEFLATE     BA6AB353 PKZIP for MVS   5.5 PKUNZIP 2.0
  _   PKZIP/TEST/PDS/DELLINK
                 DEFLATE     2F2AA610 PKZIP for MVS   5.5 PKUNZIP 2.0
  _   PKZIP/TEST/PDS/DELNUC
                 DEFLATE     2F2AA610 PKZIP for MVS   5.5 PKUNZIP 2.0
  _   PKZIP/TEST/PDS/DELNVSM
                 DEFLATE     62E3B570 PKZIP for MVS   5.5 PKUNZIP 2.0
  _   PKZIP/TEST/PDS/DELUCAT
                 DEFLATE     FF65B6F2 PKZIP for MVS   5.5 PKUNZIP 2.0
  _   PKZIP/TEST/PDS/DELVSAM
                 DEFLATE     C3593401 PKZIP for MVS   5.5 PKUNZIP 2.0
  _   PKZIP/TEST/PDS/DIAGBCS
                 DEFLATE     822BD61C PKZIP for MVS   5.5 PKUNZIP 2.0
```

Primary Commands

The primary command UP, DOWN, LEFT, and RIGHT can be entered to control the scrolling. Also the LOCATE command can be entered to position the list of files displayed to a file name beginning with the string specified. The display can also be sorted on several fields.

The format of the sort command is:

SORT *<field> <order>*

The sort field (<field>) can be one of the following:

NAME - File name.

DATE - Date zipped.

TIME - Time zipped.

ZSIZE - Compressed size.

USIZE - Uncompressed size.

RATIO - Compression ratio.

CREATED - file creation date.

REF - Last date file referenced.

The sort order (<order>) can be either:

A - For ascending.

D - For descending.

For example, to sort the display on zipped size beginning with the largest enter **SORT ZSIZE D**.

Line Commands

Once the list of files is displayed, there are several line commands that may be entered. They are entered in the left-most field next to the desired file. To execute the line commands, press ENTER. Multiple selections are allowed and will be processed in succession. To select from a list of valid commands, enter a '/' for the line command. The panel below shows the View Line Commands.

```
EssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssN
e               PKZIP for MVS 5.5 View Line Commands                        e
e Command ==>                                                               e
e                                                                           e
e Data Set: 'PKZIP.MVS.ZIP'                                                 e
e                                                                           e
e Action:                                                                   e
e                                                                           e
e    B  - Browse File               PV - Preview n Lines of File        e
e    BB - Browse Binary File        D  - Delete File                    e
e    BT - Browse Text File          I  - Display File Information       e
e    V  - View File                 X  - Extract File                   e
e    VB - View Binary File          XO - Extract with Overwrite         e
e    VT - View Text File                                                e
e                                                                           e
e Select an action and press ENTER to process                               e
e Press PF3 to return to data set list.                                     e
e                                                                           e
DssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssM
```

You can select the desired action by typing a '/' next to it. The valid commands are:

BBrowse - The selected file is extracted to a temporary file which is then displayed using ISPF browse. This option does not work for UNIX files with lower case file names. If the selected file is a VSAM file, then the file is browsed as a sequential file.

BTBrowse Text - Same as the browse command above except this will generate a –DATA_TYPE(TEXT) command for the extract to the temporary file. This is used when the file comes from another platform and/or has incomplete attributes.

BBBrowse Binary - Same as the browse command above except this will generate a –DATA_TYPE(BINARY) command for the extract to the temporary file. This is used when the file comes from another platform and/or has incomplete attributes.

DDelete - The selected file is deleted from the archive file. A confirmation panel will be displayed to confirm the delete.

IInfo - This option displays detailed information about the selected file. This display is similar to the 'I' command given on the Dataset List (3.4) display.

PVPreview Extract - This command will display the first 'n' lines of an archived file. This option can be used to view a portion of a large file without extracting the entire file. A prompt will request the number of lines to display.

VView - The selected file is extracted to a temporary file which is then displayed using ISPF view. This option does not work for VSAM files or for UNIX type files with lower case file names. Standard ISPF View commands (such as CREATE) can be used to make a copy of the file being viewed.

VTView Text - Same as the view command above except this will generate a –DATA_TYPE(TEXT) command for the extract to the temporary file. This is used when the file comes from another platform and/or has incomplete attributes.

VBView Binary - Same as the browse command above except this will generate a –DATA_TYPE(BINARY) command for the extract to the temporary file. This is used when the file comes from another platform and/or has incomplete attributes.

XExtract - The selected file is extracted from the archive file.

XOExtract with overwrite - The selected file is extracted from the archive file and will overwrite an existing file with the same name. Same as the 'X' command except the –OUTFILE_OVERWRITE(Y) command is generated.

Note: Each time a zipped file is selected for browsing or viewing, a temporary file is created. Depending on the size of the unzipped file, the temporary file may be quite large. If you are running under SMS control, SMS will attempt to find the necessary space for the large file and your terminal will be locked during that period of time.

Display Fields

There are several fields of information displayed for each file in the archive. Each field is explained in detail below:

File Name - The file name field contains the name(s) of the file(s) contained in the archive. This name can contain both upper and lower case letters. This is the only field that is repeated on each display panel. If a '+' is displayed immediately in front of the file name this indicates that that file is encrypted and any operation on that file will require a password.

Date/Time Zipped - The field contains the data and time that the file was compressed and added to the archive.

Zipped Size - This field contains the number of bytes the file contains after it was compressed. If the file attributes are incomplete or if the file was compressed in GZIP format, this field will contain 'N/A'.

Unzipped Size - This field contains the number of bytes the file contained before it was compressed. If the file attributes are incomplete or if the file was compressed in GZIP format, then this field will contain 'N/A'.

Compression Ratio - This field contains the ratio between uncompressed size and compressed size. It provides a measure of the degree of compression.

File Type - This field indicates the type of data contained in the compressed file. It can be text (TEXT) or binary (BIN).

Volume - This field indicates the volume from which the compressed file came. If it is a multi-volume file, only the first volume is displayed along with a plus sign (+) indicating there are additional volume(s).

Message - This field is used to show the last line command executed against this file. The valid displays are '*Browsed', '*Viewed', '*Info', '*Unzip', and '*Delete'.

Dsorg - This field displays the dataset organization of the compressed file. Valid entries are 'PS' for a sequential file, 'PO' for a PDS, 'VSAM', and 'PDSE' for a PDS extended file.

Record Format - Record format of the compressed file.

Record Size - Record size of the file in bytes.

Block Size - Block size of the file in bytes.

Primary Space - Amount of primary space allocation.

Secondary Space - Amount of secondary space allocation.

Allocation Units - BLKS, TRKS, or CYLS.

Directory Blocks - Number of directory blocks allocated.

Creation Date - Date the file was created.

Last-Referenced Date - Date the file was last referenced.

Compression Method - Method used to compress the file.

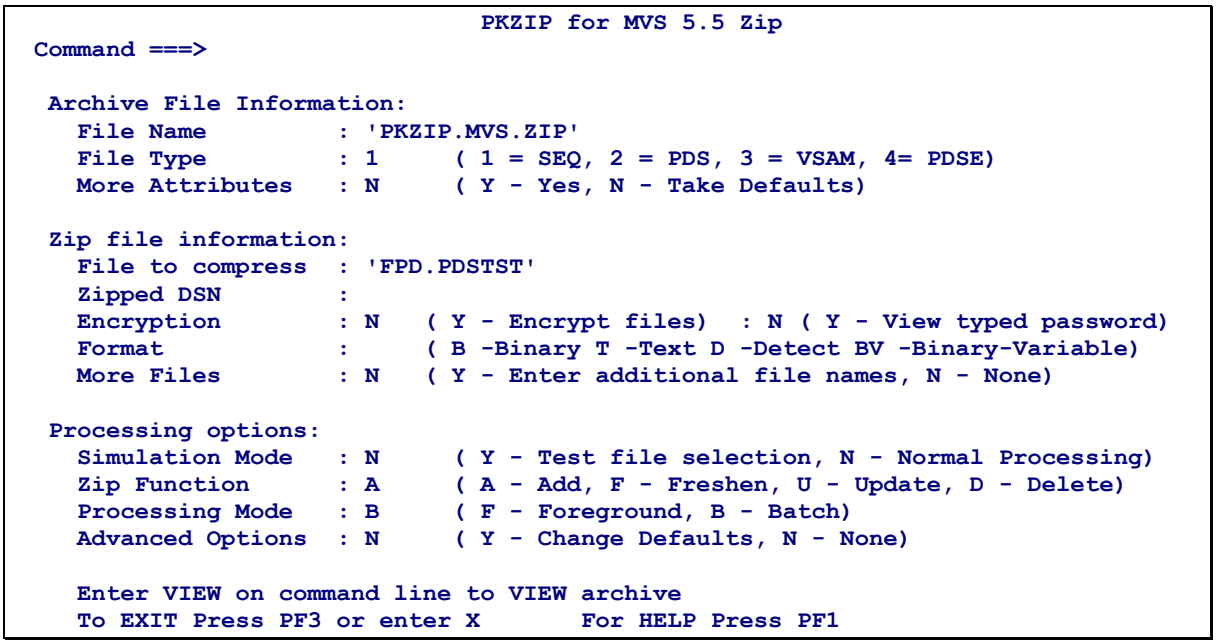
Cyclic Redundancy Check - A 32-bit field used to insure integrity of the file. This field is calculated during compression. It is re-calculated when the file is decompressed and that value is checked against the original value.

Compressed by - Program used to compress file.

Needed to Extract - Version needed to extract the file.

Zip (Option 'Z')

This option is used when a file or multiple files are to be compressed and added to a zip archive. You must enter the name of a zip archive. This file can be a new file or an existing file. Additionally, you must indicate what file(s) to compress, indicate the name of the files in the archive, and select the desired processing options. The initial panel displayed when this option is selected is shown below.


PKZIP for MVS 5.5 Zip
Command ===>
Archive File Information:
File Name : 'PKZIP.MVS.ZIP'
File Type : 1 (1 = SEQ, 2 = PDS, 3 = VSAM, 4= PDSE)
More Attributes : N (Y - Yes, N - Take Defaults)
Zip file information:
File to compress : 'FPD.PDSTST'
Zipped DSN :
Encryption : N (Y - Encrypt files) : N (Y - View typed password)
Format : (B -Binary T -Text D -Detect BV -Binary-Variable)
More Files : N (Y - Enter additional file names, N - None)
Processing options:
Simulation Mode : N (Y - Test file selection, N - Normal Processing)
Zip Function : A (A - Add, F - Freshen, U - Update, D - Delete)
Processing Mode : B (F - Foreground, B - Batch)
Advanced Options : N (Y - Change Defaults, N - None)
Enter VIEW on command line to VIEW archive
To EXIT Press PF3 or enter X For HELP Press PF1

Based upon the panel input, commands are built and included in the compress job's input (SYSIN) stream. The commands generated are fully explained in the commands chapter of this manual. The individual panel fields and their affect on processing are as follows:

Archive Name - Enter the name of the archive file. It can be in the form of DATA.SET.NAME or DATA.SET.NAME(MBR). Standard dataset naming conventions apply. Place the dataset name in single quotes ('…') to prevent using the TSO prefix as the first qualifier. This option can be turned off using the Configuration option explained earlier. This file can be a new or an existing file.

File Type - If the archive file entered above is a new file, this field is used to specify what type of archive is desired. Valid entries are '1' for a sequential file, '2' for a PDS file, '3' for a VSAM archive, and '4' for a PDS extended archive. The default is a sequential file.

More Attributes - When the archive file entered above is a new file and this field is set to 'Y', then a panel is displayed where additional allocation specifications for the archive file can be entered.

File to Compress - This field is used to specify what file(s) are to be compressed and added to the archive file. A fully qualified name can be entered or standard wildcards can be used to select multiple files. See Chapter 5. - File Selection and Name Processing for rules on file selection.

Zipped DSN - This field is used to give the compressed file a new name in the archive file. It generates a –ZIPPED_DSN command. That command is fully explained in Chapter 9. - Commands.

Encryption - A 'Y' in this field indicates that the file to be compressed should be encrypted. This will cause a panel to be displayed requesting that you enter a 1 - 200 character password to be associated with the compressed file. This field is initialized to an 'N'.

View Typed Password - Enter a 'Y' or 'N'. A 'Y' indicates that the password will be displayed while you enter it. This field is initialized to 'N'.

Format - This field indicates the file type of the file to be compressed. Valid entries are 'B' for binary, 'T' for text, 'D' for detect, and 'BV' for binary-variable. The value entered will be used to construct a –DATA_TYPE command. If the value entered is 'BV' then a –SAVE_LRECL(Y) command is also generated.

More Files - If more file selection entries are desired, enter a 'Y' in this field. Another panel will be displayed where up to 10 additional file specifications can be entered.

Simulation Mode - Specifying a 'Y' in this field will run the compress job in simulation mode. A –SIMULATE(Y) command is added to the input stream. This allows file selection and renaming operations to be verified before files are actually written. No file(s) are actually added to the archive file.

Zip Function - This field will determine the type of –ACTION command that will be generated. Valid entries are: 'A' for –ACTION(ADD), 'F' for –ACTION(FRESHEN), 'U' for –ACTION(UPDATE), and 'D' for –ACTION(DELETE).

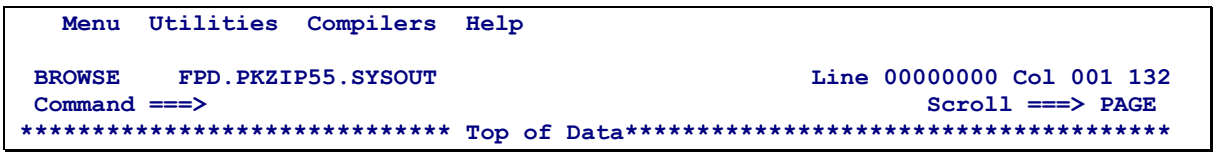
Processing Mode - Specifying a 'F' in this field will run the compress as a foreground task. Specifying a 'B' will build JCL for a batch job. The JCL will be displayed so it can reviewed and/or modified before submission. The job is submitted by the TSO SUBMIT command.

Advanced Options - Specifying a 'Y' in this field will display the current defaults for zip processing and allow them to be changed and included as commands in this extract. This is the same process described for Option 'ZD' earlier. As the options are changed they are flagged with the string '**Adv Options**'. This field is initialized to a 'N'.

After all the fields have been entered, press ENTER to process the panel and build the compress job. To display the ZIP help information, press PF1. Enter 'VIEW' as a primary command to view the current contents of the specified archive file. This VIEW option is explained above under Option 'V'.

SYSPRINT Browse (Option 'S')

This option will display the output of the last on-line operation. It is displayed in a standard ISPF browse panel. If the return code of an on-line operation exceeds the lowest allowable return code (see Configuration) then the output is automatically displayed. This option allows the browsing of the output from a run with any return code. An example of this display is shown below.



```
ZPGE001T UNZIP    STARTUP STORAGE QUERY: 24BIT=    7568K  31BIT=   31544K CACHE=
ZPLI001I PKZIP for MVS (TM), Data Compression, Version 5.5   -     09/04/02 17.56
ZPLI001I  Copyright.  2003 PKWARE of Ohio, Inc.  All Rights Reserved.
ZPLI001I  PKZIP (R) is a registered trademark of PKWARE (R), Inc.
ZPLI001I  Registered, Processor Type=9999 Processor Group=11 Serial Number=xxxxxx
ZPLI001I  OS Level: HBB7703  SP6.1.0
**************************************
* Commands generated from panel input. *
**************************************
-SUPPRESS_DYNALLOC_MSGS
-TRACE_DYNALLOC(0)
-ARCHIVE_DSN(PKZIP.MVS.ZIP)
-ACTION(EXTRACT)
-OUTFILE_DSNTYPE(SEQ)
-OUTFILE_OVERWRITE(Y)
-UNZIPPED_DSN(**,FPD.T074526.PKZIP51.TEMP)
FPD/TEST/SEQ1
-CALLMODE(ISPF)
-TRACEDALC0
-TRACE_DYNALLOC(0)
ZPAM030I INPUT  Archive opened:  PKZIP.MVS.ZIP
ZPEX002I FPD/TEST/SEQ1
```

Messages (Option ‘M’)

This option allows you to browse the HELP dataset containing the ***PKZIP for MVS***™ messages. Each message is a separate member. Select option ‘M' then do a (L)ocate on the message id and then select that member. The text of the message, any system and/or user response, and the invoking module are displayed. An example of the display for the message list is shown below.

```
  Menu  Functions  Utilities  Help
 ------------------------------------------------------------------------------
 BROWSE    QZIP.DEV.HELP                                  Row 00001 of 00290
 Command ===>                                               Scroll ===> PAGE
    Name     Prompt          Size    Created          Changed           ID
 . $CONTACT                    10   2001/05/03   2001/05/03 13:29:26   ALPHA3
 . $DEFAULT                    29   2001/05/07   2001/05/07 13:39:23   MAS01
 . $MESSAGE                    47   2001/05/03   2001/05/03 13:29:29   ALPHA3
 . $ZIPAPI                    125   2001/05/10   2001/05/10 10:58:48   MAS01
 . IKJ56228                    19   2001/05/17   2001/05/17 08:27:16   MAS01
 . ZPAM001C                    15   2001/05/03   2001/05/03 13:29:01   ALPHA3
 . ZPAM001I                    16   2001/05/03   2001/05/03 13:29:24   ALPHA3
 . ZPAM002I                    25   2001/05/03   2001/05/03 13:29:26   ALPHA3
 . ZPAM003I                    16   2001/05/03   2001/05/03 13:29:58   ALPHA3
 . ZPAM004I                    31   2001/05/03   2001/05/03 13:29:44   ALPHA3
 . ZPAM005I                    21   2001/05/03   2001/05/03 13:29:53   ALPHA3
 . ZPAM006I                    20   2001/05/03   2001/05/03 13:29:23   ALPHA3
 . ZPAM007I                    16   2001/05/03   2001/05/03 13:29:53   ALPHA3
 . ZPAM008I                    24   2001/05/03   2001/05/03 13:29:30   ALPHA3
 . ZPAM009I                    17   2001/05/03   2001/05/03 13:29:17   ALPHA3
 . ZPAM010I                    18   2001/05/03   2001/05/03 13:29:01   ALPHA3
 . ZPAM011C                    13   2001/05/03   2001/05/03 13:29:56   ALPHA3
 . ZPAM011I                    18   2001/05/03   2001/05/03 13:29:06   ALPHA3
 . ZPAM012C                    15   2001/05/03   2001/05/03 13:29:29   ALPHA3
```

If you wish to see the message text for message ZPAM914E, enter a L ZPAM9, select ZPAM914E and press ENTER. The text for the message will be displayed in an ISPF browse panel as shown below.

```
   Menu  Utilities  Compilers  Help
 sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
 BROWSE    QZIP.DEV.HELP(ZPAM914E) - 01.00          Line 00000000 Col 001 080
 Command ===>                                          Scroll ===> CSR
******************************** Top of Data ********************************
********************************************************************************
*                                                                              *
*  ZPAM914E An error occurred attempting to locate a Local Directory entry.    *
*                                                                              *
*  Explanation:     The Archive Manager was reading through the input Archive  *
*                   by using offsets and lengths according to other directory  *
*                   entries.  A Local Directory Header was expected at a       *
*                   specific offset in the file, but the eye-catcher was not   *
*                   present there.                                             *
*                                                                              *
*                   Note:  The Local Directory begins with  X'504B0304'        *
*                                                                              *
*  System Response: Processing is terminated.                                  *
*                                                                              *
*  User Response:   Determine whether the file has been truncated.             *
*                                                                              *
*  Invoking module:  <ACAMGR>                                                  *
*                                                                              *
********************************************************************************
```

License Display (Option ‘L’)

This option will display information about the ***PKZIP for MVS***™ license in effect on this processor. The initial information produced in the list is your customer number and customer name along with the current processor serial number. Licensing requirements and options are fully explained in Appendix A of this document. When this option is selected, you are asked to supply the high level qualifiers of the license dataset. Using that dataset the license data is reported and displayed. An example of the display generated by this option is shown below.

```
   Menu  Utilities  Compilers  Help
 sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
 BROWSE    PKZIP.ASILIC.TEMP                      Line 00000000 Col 001 080
 Command ===>                                          Scroll ===> CSR
******************************** Top of Data ********************************
ZPLI200I A LICENSE REPORT HAS BEEN REQUESTED ON 05/20/01 AT 7:31pm FROM CPU SERI
********************************************************************************
ZPLI200I THIS PRODUCT IS LICENSED TO  CUSTOMER # 100120127  - CUSTOMER NAME -
********************************************************************************
ZPLI200I THIS IS AN ENTERPRISE LICENSE.  ALL PRODUCT FEATURES ARE AVAILABLE ON A
ZPLI200I EXPIRATION DATE IS  01/02/2002
********************************************************************************

******************************** Bottom of Data ********************************
```

What’s New (Option ‘W’)

This option displays information about the changes included in this release of ***PKZIP for MVS***™.

Contact PKWARE (Option ‘A’)

This option will display information on how to contact PKWARE. Additionally, the display will contain information on the data that should be provided to PKWARE when reporting problems about ***PKZIP for MVS™***.

Chapter 12. - Using the Application Programming Interface (API)

ZIPAPI is a subroutine designed for the purpose of compressing/uncompressing a block of storage up to an uncompressed size of 64K from an application program.

The routine follows standard subroutine linkage conventions and therefore can be called from a high level language such as COBOL. ZIPAPI can be dynamically or statically called in any addressing mode.

Restrictions:

ZIPAPI is not re-entrant and should never be called from a subsystem environment such as CICS, IMS, or VTAM.

Calling ZIPAPI consists of three parameters:

1. ZIPPARM - Control block.

   Function: C=Compress, E=Expand.

   Source length - required to compress.

   Target length - provided by ZIPAPI.

   CRC - provided by compress, required for Expand.

   Return code.

2. Source Address - required.

3. Target Address - required.

Invoking ZIPAPI in Assembler

```
        ZIPAPIM COMPRESS,SRC=EXPDATA@,SRCLEN=EXPSIZE,TGT=CMPDATA@
```

```
Macro   Function Source Addr  Source Length  Target Addr

Macro invocation: ZIPAPIM
Function: COMPRESS
SRC:      Source Address 4 bytes
SRCLEN:   Source Length 4 bytes
TGT:      Target Address 4 bytes

Result of call:
Compressed data written to address in CMPDATA@
The following fields are updated in ZIPPARM (1st parm of CALL):
ZIP_TARGET_LEN - length of compressed data
ZIP_CRC - Cyclic Redundancy Check of input data
ZIP_RETCODE - Return Code (see explanation of return codes)
```

```
        ZIPAPIM EXPAND,SRC=CMPDATA@,TGT=EXPDATA@,CRC=ZIPCRC
```

```
Macro   Funct  Source Addr  Target Addr  CRC

Macro invocation: ZIPAPIM
Function: EXPAND
SRC: Source Address 4 bytes
TGT: Target Address 4 bytes
CRC: Cyclic Redundancy Check 4 bytes

Result of call:
Expanded data written to address in EXPDATA@
The following fields are updated in ZIPPARM (1st parm of CALL):
ZIP_TARGET_LEN - length of expanded data
ZIP_RETCODE - Return Code (see explanation of return codes)
```

ZIPPARM Copy Member

```
       ZIP_PARM        DS    0D
       ZIP_FUNCTION    DS    CL4          SERVICE REQUEST
       ZIP_COMPRESS    EQU   C'C'           COMPRESS
       ZIP_EXPAND      EQU   C'E'           EXPAND
       ZIP_SOURCE_LEN  DS    F            SOURCE LENGTH
       ZIP_TARGET_LEN  DS    F            TARGET LENGTH
       ZIP_CRC         DS    F        )   CRC
       ZIP_RETCODE     DS    F
                       DS    XL44         RESERVED
```

A sample assembler program is located in pkzip.mvs.INSTLIB(APIASM)

Invoking ZIPAPI in COBOL

```
       WORKING-STORAGE SECTION.
           COPY  ZIPPARM.
             ...
       PROCEDURE DIVISION.
             ...
      * COMPRESS A DATA BLOCK
           MOVE LOW-VALUES      TO  ZIP-PARM.
           MOVE 'C'             TO  ZIP-FUNCTION.
           MOVE 6500            TO  ZIP-SOURCE-LEN.
           MOVE 'ZIPAPI'        TO  SUBPROGRAM.
           CALL SUBPROGRAM USING ZIP-PARM, DATAIN-BLOCK, DATAZIP-BLOCK.
           IF ZIP-RETCODE  NOT EQUAL  ZERO
             PERFORM FAILED-ZIP-CALL
           ELSE
             MOVE ZIP-CRC       TO  SAVED-CRC
             ...
      * EXPAND A DATA BLOCK
           MOVE LOW-VALUES      TO  ZIP-PARM.
           MOVE 'E'             TO  ZIP-FUNCTION.
           MOVE SAVED-CRC       TO  ZIP-CRC.
           MOVE 'ZIPAPI'        TO  SUBPROGRAM.
           CALL SUBPROGRAM USING ZIP-PARM, DATAZIP-BLOCK, DATAOUT-BLOCK.
           IF ZIP-RETCODE  NOT EQUAL  ZERO
             PERFORM FAILED-ZIP-CALL
             ...

   ZIPAPI can be called as a static or dynamic subroutine.

       ZIPPARM copy member:
```

```
01  ZIP-PARM.
    02 ZIP-FUNCTION       PIC X.
      88 ZIP-COMPRESS  VALUE 'C'.
      88 ZIP-EXPAND    VALUE 'E'.
    02 FILLER             PIC X(3).
    02 ZIP-SOURCE-LEN     PIC 9(9) BINARY.
    02 ZIP-TARGET-LEN     PIC 9(9) BINARY.
    02 ZIP-CRC            PIC 9(9) BINARY.
    02 ZIP-RETCODE        PIC 9(9) BINARY.
    02 FILLER             PIC X(44).
```

A sample COBOL program is located in pkzip.mvs.INSTLIB(APICOB)

RETURN CODES

```
0  -  Call was successful
8  -  CRC failure on EXPAND call
12 -  Failure in EXPAND or COMPRESS routine
16 -  Invalid function requested
20 -  Storage allocation failure
```

Appendix A - Licensing Requirements

PKZIP MVS™ is a licensed product. Without proper licensing the product can only be used to view archives. Product features can be licensed separately as the user needs dictate. The license key will contain all of the elements necessary to validate a customer's use of ***PKZIP MVS™***.

The licensing process is comprised of several key elements that are described in the following sections.

Licensed Types

The license key will be comprised of codes to reflect the license types selected by the customer.

The list on the following page contains the types of licensing that are currently available:

Type	Description	Use
BASIC	The BASIC license type is the base line. It represents a license for which there are no restrictions, other than time. In contrast, all the other license types define restrictions within which the application is licensed and the customer is to abide.	Customer will receive a predetermined set of product features.
CAPACITY	The CAPACITY license type compares the capacity of the operating environment (as defined by the machine serial number) along with a predefined table; for instance, to assure the application is running in a machine whose computing capacity is not larger that that for which the product is licensed.	Customer will designate the serial number of the processor(s).
DEMO	A DEMO license is typically restricted to a certain time period, number of executions, or limited set of functions. These licenses may allow any of the other types of use. This license is also known as "Try and Buy" or "Supply before Buy." These terms and conditions can be an added restriction to any of the license types.	30-day trial period.
DISASTER RECOVERY	A DISASTER RECOVERY license is granted by the vendor to allow a specified product to execute under conditions defined as "disaster recovery" for a specified period of time or for a specified number of occurrences. These terms and conditions can be an added restriction to any of the license types.	Implemented with a 5-day grace period to allow the customer to contact PKWARE to update the license. The grace period will never expire on a weekend.
ENTERPRISE	An ENTERPRISE license is assigned to an enterprise; which may be comprised of multiple sites, complexes, nodes, and/or serial numbers. It is an all-encompassing license to a single entity. These terms and conditions are derived from any of the license types.	Allows a customer full access to all features of ***PKZIP MVS™*** on all systems.
FEATURES	A packaging and enablement option. An optional feature of a product can be packaged, licensed, and enabled at the discretion of the software publisher. Features can be licensed in the same manner as software products and can, therefore, be of any license type.	See product options below.
TIME-DELIMITED	Each license type is modifiable by time.	Each license will have a finite time period.

Product Features

The license key will be comprised of codes to reflect the product features selected by the customer.

The following list contains the product features currently available:

Type	Description
COMPRESSION	Allows compression of data into an archive.
DECOMPRESSION	Allows extraction of data from an archive.
SEQUENTIAL FILE HANDLER	Allows sequential file processing for archives and datasets.
PDS FILE HANDLER	Allows partition dataset processing for archives and datasets.
PDS/E FILE HANDLER	Allows enhanced partition dataset processing for archives and datasets
VSAM FILE HANDLER	Allows Virtual Storage Access Method processing for archives and datasets.
MAGNETIC TAPE/CARTRIDGE FILE HANDLER	Allows tape file processing for archives and datasets.
GIGA ZIP	Allows support for GNU zip files.
ISPF	Allows ***PKZIP MVS***™ to be called from ISPF panels.
CLI	Allows ***PKZIP MVS***™ to be called from a command.
API	Allows the use of the Application Programming Interface.
Encryption	Allows Advanced Encryption.

Licensing Environment

PKZIP MVS™ contains a series of processes that will update the current use license, allow reporting of the license information, allow conditional use of the product during a disaster recovery, and allow conditional use during a modification of the customer's physical environment.

Evaluation Period

License generation for a trial of the product allowing full use is a simple process of obtaining a key from the Sales Division. Once this process is completed PKZIP MVS will allow access to all options for a period of 30 days. At some time during this process you must contact PKWARE to obtain licensing to allow use beyond the initial period.

For Licensing, please contact the Sales Division at 937-847-2374 or email PKSALES@PKWARE.COM.

For Technical Support assistance, please contact the Product Services Division at 937-847-2687 or go to http://www.pkzip.com/support. Specify the correct high level qualifier*, unit, and volume information.

Current Use License

When you receive the license control card information from PKWARE, you will build the license dataset using the build license program (there is a sample job stream in member LICUPDAT in the Installation Dataset (INSTLIB)). By executing this job stream, the LICENSE dataset will be updated and a report will be produced that will reflect the state of ***PKZIP MVS***™ at your location.

Following is a sample of the output:

```
ZPLI200I CONTROL CARD INPUT TO THE LICENSE RECORD
*LICENSED BY PKWARE   11/22/02 FPD
55 X37C8901 104620127 PKWARE of Ohio, INC. DAYTON, OHIO
23 RT1A2217 20020102 01052B70601B
12 RT1A1331 20020102 01462A903041
14 XXOP2217 20020102 01052B70601B
73 XZZX2217 20020102 01052B70601B
18 RT562217 20020102 01052B70601B
```

```
89 1414C1EF 20020930 01052B70601B
ZPLI200I THE LICENSE RECORD HAS BEEN UPDATED FOR PKZIP ON 01/08/01 AT 1:45pm FROM CPU
SERIAL# 01052B
********************************************************************************
***********
```

Reporting

To report on the status of the license at your location, run the sample job stream in member LICPRINT in the Installation Dataset (*pkzip.mvs*.INSTLIB).

Following is a sample of the report with normal licensing:

```
ZPLI200I A LICENSE REPORT HAS BEEN REQUESTED ON 11/19/02 AT 8:35am IN QZIP.FPD.LICENSE
ZPLI200I For Technical Support assistance, please contact Product Services Division
ZPLI200I at 937-847-2687 or go to http://www.pkzip.com/support.
********************************************************************************
********
ZPLI200I THIS PRODUCT IS LICENSED TO  CUSTOMER # 000012805
ZPLI200I  - CUSTOMER NAME -   PKWARE of Ohio, Inc.
ZPLI200I CPU model 2066 with 1 online
ZPLI200I CPU serial number for CPU 0 is 01824A2066 (1824A), version code 00.
ZPLI200I Service units per second per online CPU is 5612.07
ZPLI200I Approximate total MIPS (SUs/SEC / 48.5 * #CPUs) is 115.71
ZPLI200I Central Processing Complex (CPC) Node Descriptor: CPC ND =
002066.0B1.IBM.02.00000001824A
ZPLI200I   CPC ID = 00   Type(002066) Model(0B1) Manufacturer(IBM) Plant(02) Seq
Num(00000001824A)
********************************************************************************
********
ZPLI200I COMPRESSION IS LICENSED ON THE FOLLOWING PROCESSORS
ZPLI200I SERIAL# *2824A PROCESSOR TYPE  2066 VERSION  00  WITH AN EXPIRATION DATE OF
02/28/2400
ZPLI200I DECOMPRESSION IS LICENSED ON THE FOLLOWING PROCESSORS
ZPLI200I SERIAL# *2824A PROCESSOR TYPE  2066 VERSION  00  WITH AN EXPIRATION DATE OF
02/28/2400
ZPLI200I SEQUENTIAL FILE HANDLER IS LICENSED ON THE FOLLOWING PROCESSORS
ZPLI200I SERIAL# *2824A PROCESSOR TYPE  2066 VERSION  00  WITH AN EXPIRATION DATE OF
02/28/2400
```

Following is a sample of the report with DEMO licensing:

```
ZPLI200I A LICENSE REPORT HAS BEEN REQUESTED ON 11/18/02 AT 5:25pm IN QZIP.FPD.LICENSE
ZPLI200I For Technical Support assistance, please contact Product Services Division
ZPLI200I at 937-847-2687 or go to http://www.pkzip.com/support.
********************************************************************************
********
ZPLI200I THIS PRODUCT IS LICENSED TO  CUSTOMER # 000012805 AS A DEMO
ZPLI200I  - CUSTOMER NAME -   PKWARE INC.
ZPLI200I CPU model 2066 with 1 online
ZPLI200I CPU serial number for CPU 0 is 01824A2066 (1824A), version code 00.
ZPLI200I Service units per second per online CPU is 5612.07
ZPLI200I Approximate total MIPS (SUs/SEC / 48.5 * #CPUs) is 115.71
ZPLI200I Central Processing Complex (CPC) Node Descriptor: CPC ND =
002066.0B1.IBM.02.00000001824A
ZPLI200I   CPC ID = 00   Type(002066) Model(0B1) Manufacturer(IBM) Plant(02) Seq
Num(00000001824A)
********************************************************************************
******************
ZPLI200I COMPRESSION IS LICENSED ON THE FOLLOWING PROCESSORS
ZPLI200I SERIAL# *0824A PROCESSOR TYPE  2066 VERSION  00  WITH AN EXPIRATION DATE
OF  12/01/2002
ZPLI200I DECOMPRESSION IS LICENSED ON THE FOLLOWING PROCESSORS
ZPLI200I SERIAL# *0824A PROCESSOR TYPE  2066 VERSION  00  WITH AN EXPIRATION DATE
OF  12/01/2002
ZPLI200I SEQUENTIAL FILE HANDLER IS LICENSED ON THE FOLLOWING PROCESSORS
```

```
ZPLI200I SERIAL# *0824A PROCESSOR TYPE  2066 VERSION  00  WITH AN EXPIRATION DATE
OF  12/01/2002
********************************************************************************
******************
```

Show System Information

To display hardware and software information at your location, run the sample job stream in member LICSHSYS in the Installation Dataset (*pkzip.mvs*.INSTLIB). By executing this job stream a Show System Information report will be displayed.

Following is a sample of the report:

```
ZPLI210I PKZIP for MVS (TM), DATA COMPRESSION,
Copyright. 2003 PKWARE of Ohio, Inc.  All rights reserved.
PKZIP (R) IS A REGISTERED TRADEMARK OF PKWARE (R), INC.
For Licensing, please contact the Sales Division
at 937-847-2374 or email PKSALES@PKWARE.COM
For Technical Support assistance, please contact the
Product Services Division at 937-847-2687 or go to http://www.pkzip.com/support.

Friday 11/22/2002 (2002.326)  10:17:55
CPU model 2066 with 1 online
CPU serial number for CPU 0 is 01824A2066 (1824A), version code 00.
Service units per second per online CPU is 5612.07.
Approximate total MIPS (SUs/SEC / 48.5 * #CPUs) is 115.71.

Central Processing Complex (CPC) Node Descriptor:
  CPC ND = 002066.0B1.IBM.02.00000001824A
  CPC ID = 00
  Type(002066) Model(0B1) Manufacturer(IBM) Plant(02) Seq Num(00000001824A)
JES2 z/OS 1.2 N1
DFSMS z/OS 1.3.0
```

Conditional Use

PKWARE recognizes that there may be periods where the licensing environment established by the customer is no longer valid. Circumstances such as disaster recovery processing or the installation or upgrade of new processors will affect the environment. In order to accommodate the customer, ***PKZIP MVS™*** has a process that will allow the customer to continue to use the product for a period of 5 days. During this time, error messages will be displayed on the console (as well as the printout) for each execution of ***PKZIP MVS™***. At the end of the grace period, if the license keys are not updated, the product will no longer function in any environment other than to VIEW an archive. This 5-day grace period is designed so that it will not cease to function on a weekend or the Monday following the 5-day grace period. During this period you must contact PKWARE at 1-937-847-2687 to obtain licensing to allow use beyond the conditional period.

Appendix B - Sample Jobstreams

Example 1: Zip PDS to an Archive

JCL Used

```
//SAMPZIP1 JOB  (XXXX),SAMPZIP1,
//             CLASS=B,
//             MSGCLASS=Q,
//             NOTIFY=&SYSUID,
//             REGION=8M
//*****************************************************************
//* Sample job stream to ZIP pds file "SYS1.MACLIB" to an          *
//*        archive of "PKWARE.MACLIB.ARCHIVE"                       *
//*****************************************************************
//*
//ZIP1     EXEC PGM=PKZIP,PARM='-ECHO '
//STEPLIB  DD   DISP=SHR,DSN=PKZIP.MVS.LOAD
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
-ARCHIVE_DSN(PKWARE.MACLIB.ARCHIVE)
-ACTION(ADD)
SYS1.MACLIB
/*
//
```

Resulting Output

```
ZPAM030I OUTPUT Archive opened:  PKWARE.MACLIB.ARCHIVE
ZPAM253I ADDED     File SYS1.MACLIB(ABEND)
ZPAM254I   as SYS1/MACLIB/ABEND
ZPAM255I   (DEFLATED 78%/78%)
ZPAM253I ADDED     File SYS1.MACLIB(ACB)
ZPAM254I   as SYS1/MACLIB/ACB
ZPAM255I   (DEFLATED 77%/77%)
ZPAM253I ADDED     File SYS1.MACLIB(ACBVS)
ZPAM254I   as SYS1/MACLIB/ACBVS
ZPAM255I   (DEFLATED 78%/77%)
ZPAM253I ADDED     File SYS1.MACLIB(ACI)
ZPAM254I   as SYS1/MACLIB/ACI
ZPAM255I   (DEFLATED 73%/72%)

. . . . . .  .

ZPAM253I ADDED     File SYS1.MACLIB(YREGS)
ZPAM254I   as SYS1/MACLIB/YREGS
ZPAM255I   (DEFLATED 83%/83%)
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Example 2: Zip PDS to an Archive

JCL Used

```
//SAMPZIP2 JOB  (XXXX),SAMPZIP2,
//             CLASS=B,
//             MSGCLASS=Q,
//             NOTIFY=&SYSUID,
//             REGION=8M
//*****************************************************************
//* Sample job stream to ZIP pds file "SYS1.MACLIB" to an          *
//*        archive of "PKWARE.MACLIB.ARCHIVE"                        *
//*                                                                *
//*        The second qualifier of the output member(s) will be    *
//*        changed to "MYLIB" per the ZIPPED_DSN command.          *
//*****************************************************************
//*
//ZIP2     EXEC PGM=PKZIP,PARM='-ECHO '
//STEPLIB  DD   DISP=SHR,DSN=PKZIP.MVS.LOAD
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
-ARCHIVE_DSN(PKWARE.MACLIB.ARCHIVE)
-ACTION(ADD)
-ZIPPED_DSN(*.MACLIB(*),*/MYLIB/*)
SYS1.MACLIB
/*
//
```

Resulting Output

```
ZPAM030I OUTPUT Archive opened:  PKWARE.MACLIB.ARCHIVE
ZPAM253I ADDED     File SYS1.MACLIB(ABEND)
ZPAM254I   as SYS1/MYLIB/ABEND
ZPAM255I   (DEFLATED 78%/78%)
ZPAM253I ADDED     File SYS1.MACLIB(ACB)
ZPAM254I   as SYS1/MYLIB/ACB
ZPAM255I   (DEFLATED 77%/77%)
ZPAM253I ADDED     File SYS1.MACLIB(ACBVS)
ZPAM254I   as SYS1/MYLIB/ACBVS
ZPAM255I   (DEFLATED 78%/77%)
ZPAM253I ADDED     File SYS1.MACLIB(ACI)
ZPAM254I   as SYS1/MYLIB/ACI
ZPAM255I   (DEFLATED 73%/72%)

. . . . . .

ZPAM253I ADDED     File SYS1.MACLIB(YREGS)
ZPAM254I   as SYS1/MYLIB/YREGS
ZPAM255I   (DEFLATED 83%/83%)
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Example 3: Zip VSAM KSDS to an Archive

JCL Used

```
//SAMPZIP3 JOB  (XXXX),SAMPZIP3,
//             CLASS=B,
//             MSGCLASS=Q,
//             NOTIFY=&SYSUID,
//             REGION=8M
//*****************************************************************
//* Sample job stream to ZIP VSAM KSDS file "PKWARE.SAMPLE.KSDS" to a *
//*        archive of "PKWARE.VSAMKSDS.ARCHIVE".                    *
//*                                                                 *
//*        "ARCHIVE_VOLUMES" will write the Archive to the volume  *
//*             specified.                                          *
//*                                                                 *
//*        "COMPRESSION_LEVEL(STORE)" requests NO compression.     *
//*****************************************************************
//*
//ZIP3     EXEC PGM=PKZIP,PARM='-ECHO '
//STEPLIB  DD   DISP=SHR,DSN=PKZIP.MVS.LOAD
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
-ARCHIVE_DSN(PKWARE.VSAMKSDS.ARCHIVE)
-ACTION(ADD)
-ARCHIVE_VOLUMES(PKWARE)
-COMPRESSION_LEVEL(STORE)
PKWARE.SAMPLE.KSDS
/*
//
```

Resulting output

```
ZPAM030I OUTPUT Archive opened:  PKWARE.VSAMKSDS.ARCHIVE
ZPAM253I ADDED     File PKWARE.SAMPLE.KSDS
ZPAM254I   as PKWARE/SAMPLE/KSDS
ZPAM255I   (STORED    0%/ 2%)
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Example 4: Summary View of a Dataset

JCL Used

```
//SAMVIEW1 JOB  (XXXX),SAMVIEW1,
//             CLASS=B,
//             MSGCLASS=Q,
//             NOTIFY=&SYSUID,
//             REGION=8M
//*****************************************************************
//* Sample job stream to do a summary VIEW of dataset              *
//*        "PKWARE.MACLIB.ARCHIVE".                                  *
//*****************************************************************
//*
//VIEW1    EXEC PGM=PKZIP,PARM='-ECHO '
//STEPLIB  DD   DISP=SHR,DSN=PKZIP.MVS.LOAD
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
-ARCHIVE_DSN(PKWARE.MACLIB.ARCHIVE)
-ACTION(VIEW)
/*
//
```

Resulting output

```
ZPAM030I INPUT  Archive opened:  PKWARE.MACLIB.ARCHIVE
ZPAM014I There are     1539 file(s) in the input Archive.
ZPAM012I ZIP comment: PKZIP for MVS by PKWARE of Ohio, Inc.
ZPAM013I
********************************************************************************
*********************
ZPAM015I        Length      Method            Size    Ratio    Date    Time  CRC-
32        Name
ZPAM016I --------------- ------------ --------------- ----- ---------- ----- ----
------------------------------
ZPAM017I          12,957 DEFLATE-NORM           2,856  78%  08/09/2001 11:14
36BDC0D4 SYS1/MYLIB/ABEND
ZPAM017I           6,315 DEFLATE-NORM           1,462  77%  08/09/2001 11:14
1E1A020B SYS1/MYLIB/ACB

. . . . . .
. . . . . .

ZPAM017I           2,543 DEFLATE-NORM             433  83%  08/09/2001 11:16
E0B4A859 SYS1/MYLIB/YREGS
ZPAM018I ---------------              --------------- -----
ZPAM019I     111,359,012                  17,822,596  84%
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Example 5: Summary View of a Dataset

JCL Used

```
//SAMVIEW2 JOB  (XXXX),SAMVIEW2,
//             CLASS=B,
//             MSGCLASS=Q,
//             NOTIFY=&SYSUID,
//             REGION=8M
//*****************************************************************
//* Sample job stream to do a summary VIEW of dataset              *
//*        "PKWARE.MACLIB.ARCHIVE".                                   *
//*                                                                *
//*        A request is also made to do a "BRIEF" which will       *
//*        eliminate the "CRC-32" information from being displayed.*
//*****************************************************************
//*
//VIEW2    EXEC PGM=PKZIP,PARM='-ECHO '
//STEPLIB  DD   DISP=SHR,DSN=PKZIP.MVS.LOAD
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
-ARCHIVE_DSN(PKWARE.MACLIB.ARCHIVE)
-ACTION(VIEWBRIEF)
/*
//
```

Resulting output

```
ZPAM030I INPUT  Archive opened:  PKWARE.MACLIB.ARCHIVE
ZPAM014I There are     1539 file(s) in the input Archive.
ZPAM012I ZIP comment: PKZIP for MVS by PKWARE of Ohio, Inc.
ZPAM013I
********************************************************************************
**********************
ZPAM020I        Length      Method            Size    Ratio    Date    Time
Name
ZPAM021I --------------- ------------ --------------- ----- ---------- ----- ----
------------------------------
ZPAM017I          12,957 DEFLATE-NORM           2,856  78%  08/09/2001 11:14
SYS1/MYLIB/ABEND
ZPAM017I           6,315 DEFLATE-NORM           1,462  77%  08/09/2001 11:14
SYS1/MYLIB/ACB

. . . . . .
. . . . . .

ZPAM017I           2,543 DEFLATE-NORM             433  83%  08/09/2001 11:16
SYS1/MYLIB/YREGS
ZPAM018I ---------------              --------------- -----
ZPAM019I     111,359,012                  17,822,596  84%
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Example 6: View with Detail of an Archive

JCL Used

```
//SAMVIEW3 JOB  (XXXX),SAMVIEW3,
//             CLASS=B,
//             MSGCLASS=Q,
//             NOTIFY=&SYSUID,
//             REGION=8M
//*****************************************************************
//* Sample job stream to do a VIEW with a DETAIL listing of the    *
//*        entries in "PKWARE.MACLIB.ARCHIVE".                        *
//*                                                                  *
//*        A request is also made to do a "NAME" which will        *
//*        do the listing in Data Set Name (Ascending) sequence.   *
//*****************************************************************
//*
//VIEW3    EXEC PGM=PKZIP,PARM='-ECHO '
//STEPLIB  DD   DISP=SHR,DSN=PKZIP.MVS.LOAD
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
-ARCHIVE_DSN(PKWARE.MACLIB.ARCHIVE)
-ACTION(VIEWDETAILNAME)
/*
//
```

Resulting output

```
ZPAM030I INPUT  Archive opened:  PKWARE.MACLIB.ARCHIVE
ZPAM014I There are     1539 file(s) in the input Archive.
ZPAM012I ZIP comment: PKZIP for MVS by PKWARE of Ohio, Inc.
ZPAM013I
ZPAM001I Filename: SYS1/MYLIB/ABEND
ZPAM002I File type:          TEXT
ZPAM003I Date/Time:          09-AUG-2001 11:14:34
ZPAM004I Compression Method: DEFLATE -NORMAL
ZPAM005I Compressed Size:    2,856
ZPAM006I Uncompressed Size:  12,957
ZPAM007I 32-bit CRC:         36BDC0D4
ZPAM008I Created by:         PKZIP for MVS   5.5 * - 2.x compatible
ZPAM009I Needed to extract:  PKUNZIP     2.0
ZPAM301I File Type:          NONVSAM  PDS
ZPAM302I File PDS Directory Blocks: 200
ZPAM303I File Record Format: FB
ZPAM304I File Allocation Type: TRK
ZPAM305I File Primary Space Allocated: 2245
ZPAM306I File Secondary Space Allocated: 90
ZPAM307I File Record Size: 80
ZPAM308I File Block Size: 6160
ZPAM309I File Volume(s) Used: PKWARE
ZPAM310I File Creation Date: 1998/07/27
ZPAM311I File Referenced Date: 2001/08/09
ZPAM312I File PDS Extended Directory Information:
 DIRECTORY INFORMATION FOLLOWS                          LENGTH=000004
           000000  52540647 00000000 00000000 00000000  |.................|
ZPAM313I PDS member TTRKZC: 010E07000002
ZPAM013I
ZPAM001I Filename: SYS1/MYLIB/ACB
ZPAM002I File type:          TEXT
ZPAM003I Date/Time:          09-AUG-2001 11:14:34
```

```
ZPAM004I Compression Method: DEFLATE -NORMAL
ZPAM005I Compressed Size:    1,462
ZPAM006I Uncompressed Size:  6,315
ZPAM007I 32-bit CRC:         1E1A020B
ZPAM008I Created by:         PKZIP for MVS   5.5 * - 2.x compatible
ZPAM009I Needed to extract:  PKUNZIP     2.0
ZPAM301I File Type:          NONVSAM  PDS
ZPAM302I File PDS Directory Blocks: 200
ZPAM303I File Record Format: FB
ZPAM304I File Allocation Type: TRK
ZPAM305I File Primary Space Allocated: 2245
ZPAM306I File Secondary Space Allocated: 90
ZPAM307I File Record Size: 80
ZPAM308I File Block Size: 6160
ZPAM309I File Volume(s) Used: PKWARE
ZPAM310I File Creation Date: 1998/07/27
ZPAM311I File Referenced Date: 2001/08/09
ZPAM312I File PDS Extended Directory Information:
 DIRECTORY INFORMATION FOLLOWS                              LENGTH=000004
           000000  71620002 00000000 00000000 00000000  |.................|
ZPAM313I PDS member TTRKZC: 004307000002
ZPAM013I

. . . . . .

ZPAM001I Filename: SYS1/MYLIB/YREGS
ZPAM002I File type:          TEXT
ZPAM003I Date/Time:          09-AUG-2001 11:16:24
ZPAM004I Compression Method: DEFLATE -NORMAL
ZPAM005I Compressed Size:    433
ZPAM006I Uncompressed Size:  2,543
ZPAM007I 32-bit CRC:         E0B4A859
ZPAM008I Created by:         PKZIP for MVS   5.5 * - 2.x compatible
ZPAM009I Needed to extract:  PKUNZIP     2.0
ZPAM301I File Type:          NONVSAM  PDS
ZPAM302I File PDS Directory Blocks: 200
ZPAM303I File Record Format: FB
ZPAM304I File Allocation Type: TRK
ZPAM305I File Primary Space Allocated: 2245
ZPAM306I File Secondary Space Allocated: 90
ZPAM307I File Record Size: 80
ZPAM308I File Block Size: 6160
ZPAM309I File Volume(s) Used: PKWARE
ZPAM310I File Creation Date: 1998/07/27
ZPAM311I File Referenced Date: 2001/08/09
ZPAM312I File PDS Extended Directory Information:
 DIRECTORY INFORMATION FOLLOWS                              LENGTH=000004
           000000  71690198 00000000 00000000 00000000  |...q.............|
ZPAM313I PDS member TTRKZC: 00AC09000002
ZPAM013I
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Example 7: Unzip an Archive to PDS

JCL Used

```
//SAMUNZP1 JOB  (XXXX),SAMUNZP1,
//             CLASS=B,
//             MSGCLASS=Q,
//             NOTIFY=&SYSUID,
//             REGION=8M
//*****************************************************************
//* Sample job stream to UNZIP a zipped PDS file                  *
//*        archive of "PKWARE.MACLIB.ARCHIVE" back to it's original   *
//*        content.                                               *
//*                                                               *
//*        The "FILE_EXTENSION(NAMEFILE)" will use the last       *
//*        component of the ZIPPED name as the PDS member name.   *
//*****************************************************************
//*
//UNZIP1   EXEC PGM=PKUNZIP,PARM='-ECHO '
//STEPLIB  DD   DISP=SHR,DSN=PKZIP.MVS.LOAD
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
-FILE_EXTENSION(NAMEFILE)
-ARCHIVE_DSN(PKWARE.MACLIB.ARCHIVE)
/*
//
```

Resulting output

```
ZPAM030I INPUT  Archive opened: PKWARE.MACLIB.ARCHIVE
ZPEX002I SYS1/MACLIB/ABEND
ZPEX003I Extracted to SYS1.MACLIB(ABEND)
ZPEX002I SYS1/MACLIB/ACB
ZPEX003I Extracted to SYS1.MACLIB(ACB)

. . . . . .

ZPEX002I SYS1/MACLIB/YREGS
ZPEX003I Extracted to SYS1.MACLIB(YREGS)
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Example 8: Unzip an Archive to PDS

JCL Used

```
//SAMUNZP2 JOB  (XXXX),SAMUNZP2,
//             CLASS=B,
//             MSGCLASS=Q,
//             NOTIFY=&SYSUID,
//             REGION=8M
//*****************************************************************
//* Sample job stream to UNZIP a zipped PDS file                  *
//*        archive of "PKWARE.MACLIB.ARCHIVE" back to it's original   *
//*        content.                                               *
//*                                                               *
//*        The "FILE_EXTENSION(NAMEFILE)" will use the last       *
//*        component of the ZIPPED name as the PDS member name.   *
//*                                                               *
//*        The "UNZIPPED_DSN" is being used to change the HLQ of  *
//*        the file.  While it was ZIPPED as "SYS1" it will be    *
//*        UNZIPPED as "SYS2".                                    *
//*****************************************************************
//*
//UNZIP2   EXEC PGM=PKUNZIP,PARM='-ECHO '
//STEPLIB  DD   DISP=SHR,DSN=PKZIP.MVS.LOAD
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
-UNZIPPED_DSN(SYS1,SYS2)
-FILE_EXTENSION(NAMEFILE)
-ARCHIVE_DSN(PKWARE.MACLIB.ARCHIVE)
/*
//
```

Resulting output

```
ZPAM030I INPUT  Archive opened: PKWARE.MACLIB.ARCHIVE
ZPEX002I SYS1/MACLIB/ABEND
ZPEX003I Extracted to SYS2.MACLIB(ABEND)
ZPEX002I SYS1/MACLIB/ACB
ZPEX003I Extracted to SYS2.MACLIB(ACB)

. . . . . .

ZPEX002I SYS1/MACLIB/YREGS
ZPEX003I Extracted to SYS2.MACLIB(YREGS)
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Example 9: Unzip an Archive to VSAM KSDS

JCL Used

```
//SAMUNZP3 JOB  (XXXX),SAMUNZP3,
//             CLASS=B,
//             MSGCLASS=Q,
//             NOTIFY=&SYSUID,
//             REGION=8M
//*****************************************************************
//* Sample job stream to UNZIP a zipped VSAM file                 *
//*        archive of "PKWARE.VSAMKSDS.ARCHIVE" back to it's original *
//*        VSAM structure.                                        *
//*****************************************************************
//*
//UNZIP3   EXEC PGM=PKUNZIP,PARM='-ECHO '
//STEPLIB  DD   DISP=SHR,DSN=PKZIP.MVS.LOAD
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
-ARCHIVE_DSN(PKWARE.VSAMKSDS.ARCHIVE)
/*
//
```

Resulting output

```
ZPAM030I INPUT  Archive opened:  PKWARE.VSAMKSDS.ARCHIVE
ZPEX002I PKWARE/SAMPLE/KSDS
ZPEX003I Extracted to PKWARE.SAMPLE.KSDS
ZPMT002I PKZIP processing complete.      RC=00000000        0(Dec)
```

Appendix C - Invoking PKZIP/PKUNZIP from a Program

PKZIP and PKUNZIP can be called (invoked dynamically) from other programs using standard calling conventions. Because PKZIP/PKUNZIP adheres to IBM's standard linkage conventions, the passing of parameters is accomplished as described in the language reference manuals. Because standard conventions are used, the return code is passed back to your program.

PKZIP for MVS™ is specifically designed to compress and decompress datasets. If you need to process a stream or block of data then you should use the ***PKZIP for MVS™*** API. The API is designed specifically for inclusion into user written programs and offers the programmer the flexibility to process data from any source. Further information on the ***PKZIP for MVS™*** API can be obtained from your supplier.

When designing and writing your program, the following items should be considered:

1. PKZIP/PKUNZIP adheres to standard IBM linkage conventions.

2. Return codes are passed back to your program via the standard conventions of the language used (for example, in Assembly language, return codes are passed back via register 15).

3. The configuration file will be used if available.

4. The parameters for PKZIP/PKUNZIP can be set by passing one or more parameters separated by a blank in the pass area supplied by the calling program. The length of the pass area is defined by the calling program. The length passed should reflect actual lengths of parameters passed to avoid excess parsing of unused storage (which may result in errors).

5. Specify the –NOSYSIN command in the pass parameter area if you are not planning to use SYSIN for command input. Note that problems can occur if PKZIP attempts to open the SYSIN dataset after it has already been opened by the calling program.

6. When using call parameters for command input, the following commands (if used) should be contained within the first 256 bytes of the command stream and must be upper case: –NOSYSIN, –DM, –ECHO, –NOECHO, –VERBOSE, –QUIET.

7. It is your responsibility to allocate and free DD statements referenced in the run request (such as –INFILE_DD).

8. Output is written to SYSPRINT.

9. To load PKZIP/PKUNZIP, the load library must in the link list, JOBLIB, or STEPLIB.

One example of using a program to call PKZIP would be to set up a special setting in the parameters and to analyze return codes or to change the return. The included sample Assembly program may be a good starting place to build such a pre-processor.

Sample source programs (and their JCL) are available In the dataset "pkzip.mvs.INSTLIB". These samples demonstrate the dynamical calling of PKZIP by passing parameters and using SYSIN. The following members are supplied in "pkzip.mvs.INSTLIB":

CALLASMJ	JCL to compile, link and executes the Assembly sample CALLZIPA.
CALLZIPA	Sample Assembly source program to call PKZIP.
CALLCOBJ	JCL to compile, link and executes the COBOL sample CALLZIC.
CALLZIPC	Sample COBOL program source to call PKZIP.
CALLPLIJ	JCL to compile, link and executes the PL/I sample CALLZIPP.
CALLZIPP	Sample PL/I source program to call PKZIP.
CALLREXJ	JCL to run the REXX sample CALLZIPR.
CALLZIPR	Sample REXX source program to call PKZIP.

CALLZIPC Sample Assembly Source to Call PKZIP

```
CALLZIPA TITLE 'CALLZIPA - PKZIP for MVS PREPROCESSOR'
*********************************************************************
*                                                                   *
* PKZIP for MVS (TM), DATA COMPRESSION, VERSION 5.5                     *
* COPYRIGHT.  2003 PKWARE of Ohio, INC.  ALL RIGHTS RESERVED.        *
*                                                                   *
*********************************************************************
*                                                                   *
*NAME:          CALLZIPA                                           *
*PURPOSE: Sample Assembly Program to fetch and call PKZIP          *
*      Steps:     - Pass Modified Parms                             *
*                 - Examine The Return Code On Exit                 *
*                                                                   *
* This sample Assembly programs demonstrates the ability to fetch   *
* and call PKZIP or PKUNZIP from an application program as a        *
* dynamic call (i.e., PKZIP and PKUNZIP are NOT linked into the     *
* program).  There are three main variables used in calling PKZIP.  *
* First is the program variable (PKZIPEP) containing the name of    *
* program to call.  The second variable is the parameters pass area *
* which was passed from the JCL.  You can build your own variable   *
* and use it as the PARMS to PKZIP by loading its address in EXECPARM.*
* The third variable to be concerned about is the return code pass  *
* back from PKZIP/PKUNZIP program. This can be examine for other    *
* processing or verification.                                       *
*                                                                   *
* This Example is passing the parameter '- SHOW_SETTINGS'.  If      *
* -NOSYSIN is also passed ('- SHOW_SETTINGS - NOSYSIN'), PKZIP would *
* not read other parameters from SYSIN.  This example will read     *
* parameters from //SYSIN.                                          *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
*********************************************************************
* REGISTERS EQUATES AND USAGE FOR PROGRAM CALLZIPA
*********************************************************************
*                   ENTRY            IN PROGRAM              RETURN
*R0       EQU   0   IRRELEVANT      MACRO WORK              RESTORED
*R1       EQU   1   ADDR OF PARMS   MACRO WORK              RESTORED
*R2       EQU   2   IRRELEVANT                              RESTORED
*R3       EQU   3   IRRELEVANT      COMMAND BUFFER MAPPING  RESTORED
*R4       EQU   4   IRRELEVANT                              RESTORED
*R5       EQU   5   IRRELEVANT                              RESTORED
*R6       EQU   6   IRRELEVANT                              RESTORED
*R7       EQU   7   IRRELEVANT                              RESTORED
*R8       EQU   8   IRRELEVANT                              RESTORED
*R9       EQU   9   IRRELEVANT                              RESTORED
*R10      EQU   10  IRRELEVANT                              RESTORED
*R11      EQU   11  IRRELEVANT                              RESTORED
*R12      EQU   12  IRRELEVANT      *** BASE REGISTER ***   RESTORED
*R13      EQU   13  O/S SAVEAREA    LOCAL SAVE/WORK AREAS   RESTORED
*R14      EQU   14  RETURN ADDR     STANDARD RETURN ADDRESS RESTORED
*R15      EQU   15  EP ADDR         MACRO RET CODES         RET CODE
*
*********************************************************************
* *** ESTABLISH STANDARD MODULE PROLOG.                             *
*********************************************************************
*
CALLZIPA CSECT
CALLZIPA RMODE 24
```

```
CALLZIPA AMODE 31
*
** ESTABLISH BASIC LINKAGE
*
         USING CALLZIPA,R15               TEMPORARY ADDRESSING
         SAVE  (14,12)
         LA    R14,SAVEAREA
         ST    R14,8(R13)                 SAVE BACK OUR SAVE AREA
         ST    R13,SAVEAREA+4             KEEP CALLER'S SAVE AREA
         LA    R13,SAVEAREA               LOCAL SAVE AREA
         ST    R1,EXECPARM                KEEP EXEC PARM ADDRESS

         LR    R12,R15                    ESTABLISH
         DROP  R15                         ADDRESSABILITY
         USING CALLZIPA,R12                 USING R12
*
** LOAD THE PKZIP PROGRAM INTO STORAGE AND BRANCH-ENTER IT
*
         L     R3,=A(PKZIPEP)           LOAD ADDRESS OF PROGRAM TO CALL
*                                       FETCH THE PROGRAM
         LOAD  EPLOC=(R3)
         LR    R15,R0                   HAVE EP ADDRESS
         L     R1,EXECPARM              EXEC PGM=...,PARM='...'
         BASR  R14,R15
*
** PLACE RETURN CODE EXAMINATION CODE BELOW
*
*        C     R15,=F'4'                  SAMPLE CHECK FOR RC4
*
GOBACK   L     R13,SAVEAREA+4             GET CALLER'S SAVE AREA BACK
         RETURN (14,12),RC=(15)
SAVEAREA DC    18F'0'
EXECPARM DS    F                          PASSED REG1
PKZIPEP  DC    CL8'PKZIP'
         LTORG
******** ***** ************************** ********************
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END CALLZIPA.
```

CALLZIPA Sample COBOL Source to Call PKZIP

```
000100 ID  DIVISION.
000200 PROGRAM-ID. CALLZIPC.
000300* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
000400* PKZIP for MVS (TM), DATA COMPRESSION, VERSION 5.5             *
000500* COPYRIGHT.  2003 PKWARE of Ohio, INC.  ALL RIGHTS RESERVED.*
000600*                                                               *
000700*                                                               *
000800* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
000900 ENVIRONMENT DIVISION.
001000 INPUT-OUTPUT SECTION.
001100* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
001200* PKZIP for MVS (TM), DATA COMPRESSION, VERSION 5.5            *
001300* COPYRIGHT.  2003 PKWARE of Ohio, INC.  ALL RIGHTS RESERVED.  *
001400*                                                               *
001500* Program: CALLZIPC                                            *
001600*                                                               *
001700* This sample COBOL programs demonstrates the ability to call *
001800* PKZIP or PKUNZIP from an application program as a dynamic   *
001900* call (i.e., PKZIP and PKUNZIP are NOT linked into the       *
002000* program).  There are two main variables used in calling     *
002100* PKZIP.  First is the program variable which contains the    *
002200* name of program to call.  By making it a variable forces a  *
002300* dynamic call to PKZIP.  The second variable is the          *
002400* parameters pass area in the LINKAGE SECTION.                 *
002500* The length is left up to the user, but the first two bytes  *
002600* must be a binary length of the pass  area.  See CALL-PARMS  *
002700* variables.                                                   *
002800*                                                               *
002900* This example is using the pass area of 100 bytes. This      *
003000* Example is passing the parameter '- SHOW_SETTINGS'.  If     *
003100* -NOSYSIN also passed ('- SHOW_SETTINGS - NOSYSIN'), PKZIP   *
003200* would not read other parameters from SYSIN.  n This example *
003300* it will read parameters from //SYSIN.                        *
003400*                                                               *
003500* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
003600 FILE-CONTROL.
003700 DATA DIVISION.
003800 FILE SECTION.
003900 WORKING-STORAGE SECTION.
004000 01  CALL-PROGRAM           PIC X(8).
004100 01  CALL-PARMS.
004200     02 CALL-PARM-LENGTH    PIC 9(3) BINARY VALUE 100.
004300     02 CALL-PARM-DATA      PIC X(100).
004400 LINKAGE SECTION.
004500 01  PARM-CARD.
004600     02  PARM-LENGTH        PIC 9(4) BINARY.
004700     02  PARM-DATA          PIC X(80).
004800 PROCEDURE DIVISION USING PARM-CARD.
004900     DISPLAY 'ABOUT TO CALL PKZIP'.
005000* Move Of Program Name To Variable Forces Dynamic Call.
005100     MOVE 'PKZIP'           TO CALL-PROGRAM.
005200*
005300* Set the PARM Variable field  used by PKZIP.
005400* If you do not want to read any paramters in PKZIP add -NOSYSIN
005500*
005600     MOVE '-SHOW_SETTINGS'  TO CALL-PARM-DATA.
005700     CALL CALL-PROGRAM USING CALL-PARMS.
005800     DISPLAY 'PKZIP COMPLETE RC=' RETURN-CODE.
005900     STOP RUN..
```

CALLZIPP Sample PL/I Source to Call PKZIP

```
CALLZIP: PROCEDURE OPTIONS(MAIN);
 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
 /* PKZIP for MVS (TM), DATA COMPRESSION, VERSION 5.5             */
 /* COPYRIGHT.  2003 PKWARE of OHIO, INC.  ALL RIGHTS RESERVED.*/
 /*                                                               */
 /*                                                               */
 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
 /* Program: CALLZIPP                                             */
 /*                                                               */
 /* This sample PL/I programs demonstrates the ability to call*/
 /* PKZIP or PKUNZIP from an application program as a dynamic */
 /* call (i.e., PKZIP and PKUNZIP are NOT linked into the     */
 /* program).  There are three main variables used in calling */
 /* PKZIP.  First is the program variable PKZIP which contains*/
 /* the name of program to call.  The second variable is the  */
 /* parameters pass area MY_PARM with the length being left up*/
 /* to the user.  This example is using the pass area of 30   */
 /* bytes.                                                        */
 /* This Example is passing the parameter '- SHOW_SETTINGS'.  */
 /* If -NOSYSIN is also passed ('- SHOW_SETTINGS - NOSYSIN')  */
 /* PKZIP, would not read other parameters from SYSIN.  This  */
 /* example it will read parameters from //SYSIN.             */
 /*                                                               */
 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

 /* define the PKZIP variable return code save area           */
   DECLARE MY_RETURN_CODE FIXED BINARY(15);

 /* define PKZIP as an external variable with options         */
   DECLARE PKZIP ENTRY EXTERNAL('PKZIP') OPTIONS(RETCODE,ASSEMBLER);

 /* define the area for the parameters that are passed to PKZIP.*/
 /*      The length is left to user */
   DECLARE MY_PARMS CHAR(30) VARYING;

   DECLARE PLIRETV BUILTIN;
   DISPLAY ('Invoking Pkzip');

 /*  Set calling paramters and call PKZIP                        */
   MY_PARMS ='-SHOW_SETTINGS'; /* Set the PARMS for PKZIP        */
   FETCH PKZIP;                /* Dynamically fetch PKZIP        */
   CALL PKZIP(MY_PARMS);       /* Call PKZIP passing the PARMS   */
   MY_RETURN_CODE=PLIRETV;     /* save the Return code from PKZIP */
 /*                                                               */
   DISPLAY ('RETURNED FROM PKZIP RC=' || MY_RETURN_CODE);
 END;.
```

CALLZIPR Sample REXX Source to Call PKZIP

```
/* REXX  ---------------------------------------------------------*/
/* NAME:    PKZZIP                                                */
/* PARMS:   PKZIPLoad - The name of the current PKZIP Load data set. */
/*          ARGPARMS - PKZIP passed parameters enclosed in quotes    */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* PKZIP for MVS (TM), DATA COMPRESSION, VERSION 5.5                    */
/* COPYRIGHT.  2003 PKWARE of Ohio, INC.  ALL RIGHTS RESERVED.        */
/*                                                                   */
/*                                                                   */
/* This sample REXX program demonstrates the ability to call         */
/* PKZIP or PKUNZIP from an application program as a dynamic         */
/* call.  There are three main variables used in calling PKZIP       */
/* First is the current PKZIP Load Library (PKZIPLoad) where         */
/* PKZIP can be found.  The second variable is the parameter         */
/* pass area CALLPARMS where the parameters for PKZIP are            */
/* passed to PKZIP program. This example uses an input argument      */
/* to REXX to load the PARMS.  This Example is passing the           */
/*  parameter  '-ECHO -VERBOSE -SHOW_SETTINGS'.                      */
/* If -NOSYSIN is also passed ('- SHOW_SETTINGS - NOSYSIN'),         */
/* PKZIP would not read other parameters from SYSIN.  This           */
/* example will read parameters from //SYSIN.                        */
/*                                                                   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*-------------------------------------------------------------------*/
ARG PKZIPLoad ARGPARMS

/* display the passed parameters        */
 LINE = 'REXX Sample to call PKZIP is starting'
 SAY  LEFT(LINE,80)
 LINE = 'REXX Sample PKZIP Load lib is =' PKZIPLoad
 SAY  LEFT(LINE,80)
 LINE = 'REXX Sample parameters =' ARGPARMS
 SAY  LEFT(LINE,80)

 CALLPARMS = Strip(ARGPARMS,'B',"'")   /* strip the quotes from PARMS */


 /* If running from TSO other dataset will have to allocated  */
    /*     "ALLOC FI(SYSPRINT) DA('*') SHR REUSE"   */
    /*     "ALLOC FI(SYSABEND) DA('*') SHR REUSE"   */
    /*     "ALLOC FI(SYSIN) DA('*') SHR REUSE"      */



  /* Could set other parameters such as CALLPARMS = '-SHOW_SETTINGS' */

  "Call '"PKZIPLoad"(PKZIP)' '"CALLPARMS"'"

 LINE = 'REXX Sample to call PKZIP ended with Return=' RC
 SAY  LEFT(LINE,80)

 /* If running from TSO you will need to reset all the file assignments */
         /*  "FREE FI(SYSIN)"     */
         /*  "FREE FI(SYSABEND)"  */
         /*  "FREE FI(SYSPRINT)"  */
.
```

Appendix D - 3480/3490 Installation JCL (COPYCART)

```
//<jobcard>
//*
//*********************************************************************
//*                                                                   *
//*  All lines with '<===='; "lowercase" values will require          *
//*  review & change.                                                 *
//*                                                                   *
//*  In ISPF use the  < CHANGE ALL > command to edit                  *
//*  the lower case parameter selections to the value                 *
//*  you select, for instance if UNIT=SYSDA is valid                  *
//*  for JCL enter < CHANGE ALL sysda SYSDA > to                      *
//*  replace all occurrences in this member.                          *
//*                                                                   *
//*  CHANGE ALL:                                                      *
//*    Edit the Job Card as needed.                                   *
//*                                                                   *
//*    pkzip.mvs - to the ALIAS for PKZIP MVS files.                  *
//*                                                                   *
//*    disk      - to the UNIT type for PDS files.                    *
//*                                                                   *
//*    sysda     - to the UNIT type for temporary files.              *
//*                                                                   *
//*    pkzip1    - to the Volume Serial Number of the install tape.   *
//*                                                                   *
//*    tape      - to the UNIT type for tape.                         *
//*                                                                   *
//*    volume    - to the VOLUME for the PDS files.                   *
//*                                                                   *
//*********************************************************************
//*
//*********************************************************************
//*        ==>RESTORE "PKZIP.MVS.CEXEC" TO CUSTOMERS DASD<==          *
//*********************************************************************
//JS010    EXEC PGM=IEBCOPY
//*
//SYSUT1   DD   DSN=PKWARE.MVS.CEXEC,
//             UNIT=tape,LABEL=(,SL),                        <===
//             DISP=OLD,VOL=(,RETAIN,,,SER=pkzip1)           <===
//*
//SYSUT2   DD   DSN=pkzip.mvs.CEXEC,                         <===
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(2,1,5)),
//             UNIT=disk,                                    <===
//             VOL=SER=volume                                <===
//*
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL,(5,5))                 <===
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL,(5,5))                 <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD   *
   COPY INDD=SYSUT1,OUTDD=SYSUT2
/*
//*
//*********************************************************************
//*        ==>RESTORE "PKZIP.MVS.HELP" TO CUSTOMERS DASD<==           *
//*********************************************************************
```

```
//JS020    EXEC PGM=IEBCOPY
//*
//SYSUT1   DD   DSN=PKWARE.MVS.HELP,
//             VOL=(,RETAIN,REF=*.JS010.SYSUT1),
//             UNIT=(tape,,DEFER),LABEL=(2,SL),            <===
//             DISP=OLD
//*
//SYSUT2   DD   DSN=pkzip.mvs.HELP,                        <===
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(3,1,70)),
//             UNIT=disk,                                  <===
//             VOL=SER=volume                              <===
//*
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL,(5,5))               <===
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL,(5,5))               <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD   *
   COPY INDD=SYSUT1,OUTDD=SYSUT2
/*
//*
//*******************************************************************
//*        ==>RESTORE "PKZIP.MVS.INSTLIB" TO CUSTOMERS DASD<==       *
//*******************************************************************
//JS030    EXEC PGM=IEBCOPY
//*
//SYSUT1   DD   DSN=PKWARE.MVS.INSTLIB,
//             VOL=(,RETAIN,REF=*.JS020.SYSUT1),
//             UNIT=(tape,,DEFER),LABEL=(3,SL),            <===
//             DISP=OLD
//*
//SYSUT2   DD   DSN=pkzip.mvs.INSTLIB,                     <===
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(2,1,20)),
//             UNIT=disk,                                  <===
//             VOL=SER=volume                              <===
//*
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL,(5,5))               <===
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL,(5,5))               <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD   *
   COPY INDD=SYSUT1,OUTDD=SYSUT2
/*
//*
//*******************************************************************
//*        ==>RESTORE "PKZIP.MVS.LOAD" TO CUSTOMERS DASD<==          *
//*******************************************************************
//JS040    EXEC PGM=IEBCOPY
//*
//SYSUT1   DD   DSN=PKWARE.MVS.LOAD,
//             VOL=(,RETAIN,REF=*.JS030.SYSUT1),
//             UNIT=(tape,,DEFER),LABEL=(4,SL),            <===
//             DISP=OLD
//*
//SYSUT2   DD   DSN=pkzip.mvs.LOAD,                        <===
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(5,1,20)),
//             UNIT=disk,                                  <===
//             VOL=SER=volume                              <===
//*
```

```
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD   *
   COPY INDD=SYSUT1,OUTDD=SYSUT2
/*
//*
//*******************************************************************
//*        ==>RESTORE "PKZIP.MVS.MACLIB" TO CUSTOMERS DASD<==       *
//*******************************************************************
//JS050    EXEC PGM=IEBCOPY
//*
//SYSUT1   DD   DSN=PKWARE.MVS.MACLIB,
//             VOL=(,RETAIN,REF=*.JS040.SYSUT1),
//             UNIT=(tape,,DEFER),LABEL=(5,SL),             <===
//             DISP=OLD
//*
//SYSUT2   DD   DSN=pkzip.mvs.MACLIB,                       <===
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(1,1,5)),
//             UNIT=disk,                                   <===
//             VOL=SER=volume                               <===
//*
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD   *
   COPY INDD=SYSUT1,OUTDD=SYSUT2
/*
//*
//*******************************************************************
//*        ==>RESTORE "PKZIP.MVS.SPKZCLIB" TO CUSTOMERS DASD<==     *
//*******************************************************************
//JS060    EXEC PGM=IEBCOPY
//*
//SYSUT1   DD   DSN=PKWARE.MVS.SPKZCLIB,
//             VOL=(,RETAIN,REF=*.JS050.SYSUT1),
//             UNIT=(tape,,DEFER),LABEL=(6,SL),             <===
//             DISP=OLD
//*
//SYSUT2   DD   DSN=pkzip.mvs.SPKZCLIB,                     <===
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(2,1,5)),
//             UNIT=disk,                                   <===
//             VOL=SER=volume                               <===
//*
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD   *
   COPY INDD=SYSUT1,OUTDD=SYSUT2
/*
//*
//*******************************************************************
//*        ==>RESTORE "PKZIP.MVS.SPKZMLIB" TO CUSTOMERS DASD<==     *
//*******************************************************************
//JS070    EXEC PGM=IEBCOPY
```

```
//*
//SYSUT1   DD   DSN=PKWARE.MVS.SPKZMLIB,
//              VOL=(,RETAIN,REF=*.JS060.SYSUT1),
//              UNIT=(tape,,DEFER),LABEL=(7,SL),            <===
//              DISP=OLD
//*
//SYSUT2   DD   DSN=pkzip.mvs.SPKZMLIB,                     <===
//              DISP=(NEW,CATLG,DELETE),
//              SPACE=(CYL,(2,1,5)),
//              UNIT=disk,                                  <===
//              VOL=SER=volume                              <===
//*
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD   *
   COPY INDD=SYSUT1,OUTDD=SYSUT2
/*
//*
//*******************************************************************
//*        ==>RESTORE "PKZIP.MVS.SPKZPLIB" TO CUSTOMERS DASD<==      *
//*******************************************************************
//JS080    EXEC PGM=IEBCOPY
//*
//SYSUT1   DD   DSN=PKWARE.MVS.SPKZPLIB,
//              VOL=(,RETAIN,REF=*.JS070.SYSUT1),
//              UNIT=(tape,,DEFER),LABEL=(8,SL),            <===
//              DISP=OLD
//*
//SYSUT2   DD   DSN=pkzip.mvs.SPKZPLIB,                     <===
//              DISP=(NEW,CATLG,DELETE),
//              SPACE=(CYL,(2,1,20)),
//              UNIT=disk,                                  <===
//              VOL=SER=volume                              <===
//*
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD *
   COPY INDD=SYSUT1,OUTDD=SYSUT2
/*
//*
//*******************************************************************
//*        ==>RESTORE "PKZIP.MVS.SPKZTLIB" TO CUSTOMERS DASD<==      *
//*******************************************************************
//JS090    EXEC PGM=IEBCOPY
//*
//SYSUT1   DD   DSN=PKWARE.MVS.SPKZTLIB,
//              VOL=(,RETAIN,REF=*.JS080.SYSUT1),
//              UNIT=(tape,,DEFER),LABEL=(9,SL),            <===
//              DISP=OLD
//*
//SYSUT2   DD   DSN=pkzip.mvs.SPKZTLIB,                     <===
//              DISP=(NEW,CATLG,DELETE),
//              SPACE=(CYL,(2,1,5)),
//              UNIT=disk,                                  <===
//              VOL=SER=volume                              <===
//*
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL,(5,5))                <===
```

```
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL,(5,5))              <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD   *
   COPY INDD=SYSUT1,OUTDD=SYSUT2
/*
//*
//*******************************************************************
//*        ==>RESTORE "PKZIP.MVS.SMPE.MCS" TO CUSTOMERS DASD<==      *
//*******************************************************************
//JS100    EXEX PGM=IEBGENER
//*
//SYSUT1   DD   DSN=PKWARE.MVS.SMPE.MCS,
//             VOL=(,RETAIN, REF=*.JS090.SYSUT1),
//             UNIT=(tape,,DEFER) ,LABEL=(10,SL),            <===
//             DISP=OLD
//*
//SYSUT2   DD   DSN=pkzip.mvs.SMPE.MCS,                      <===
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(2,1)),
//             UNIT=disk,                                    <===
//             VOL=SER=volume                                <===
//*
//SYSUT3   DD   UNIT=sysda,SPACE=(CYL, (5,5))                <===
//SYSUT4   DD   UNIT=sysda,SPACE=(CYL, (5,5))                <===
//*
//SYSPRINT DD   SYSOUT=*
//*
//SYSIN    DD   DUMMY
//*
//
```

Glossary

This glossary provides definitions for items that may have been referenced in the **PKZIP®** documentation. It is not meant to be exhaustive. There are excellent source of documentation for computing terms on the Internet. For example:

IBM's Terminology Web Site	http://www.networking.ibm.com/nsg/nsgmain.htm

Absolute Path Name

A string of characters that is used to refer to an object, starting at the highest level (or root) of the directory hierarchy. The absolute path name must begin with a slash (/), which indicates that the path begins at the root. This is in contrast to a Relative Path Name.

Access Method

A technique that is used to read a record from, or to write a record into, a file. Usually either: SAM (Sequential Access Method - where records are processed one after another in the order in which they appear in the file), or random (the individual records can be processed in any order) such as VSAM).

AES

The Advanced Encryption Standard is the official US Government encryption standard for customer data.

Alternate Index

An index of a file based on a key different from the base. It allows the file to be processed in a secondary key order.

American Standard Code for Information Interchange

The ASCII code (American Standard Code for Information Interchange) was developed by the American National Standards Institute for information exchange among data processing systems, data communications systems, and associated equipment, and is the standard character set used on MS-DOS and UNIX-based operating systems. In a ZIP archive, ASCII is used as the normal character set for compressed text files. The ASCII character set consists of 7-bit control characters and symbolic characters, plus a single parity bit. Since ASCII is used by most microcomputers and printers, text-only files can be transferred easily between different kinds of computers and operating systems. While ASCII code does include characters to indicate backspace, carriage return, etc., it does not include accents and special letters that are not used in English. To accomodate those special characters, Extended ASCII has additional characters (128-255). Only the first 128 characters in the ASCII character set are standard on all systems. Others may be different for a given language set. It may be necessary to create a different translation tables (see Translation Table) to create standard translation between ASCII and other character sets.

American National Standards Institute (ANSI)

An organization sponsored by the Computer and Business Equipment Manufacturers Association for establishing voluntary industry standards.

ANSI

See American National Standards Institute shown above.

APAR

See Authorized Program Analysis Report.

API

See Application Programming Interface.

Application Programming Interface

An interface between the operating system (or systems-related program) that allows an application program written in a high-level language to use specific data or services of the operating system or the program. The API also allows you to develop an application program written in a high-level language to access PKZIPxxx data and/or functions of the PKZIPxxx system.

Application System/400 (iSeries)

One of a family of general purpose systems with a single operating system, Operating System/400, that provides application portability across all models.

Archive

(1) The act of transferring files from the computer into a long-term storage medium. Archived files are often compressed to save space.
(2) An individual file or group of files which must be extracted and decompressed in order to be used.
(3) A file stored on a computer network, which can be retrieved by a file transfer program (FTP) or other means.
(4) The PKZIP file that holds the compressed/zipped datafile.

ASCII

See American Standard Code for Information Interchange.

Authorized Program Analysis Report (APAR)

A request for correction of a defect in a current release of an IBM-supplied program.

Bandwidth

The capacity of a communications line, normally expressed in bits per second (bps). A lack of bandwidth is one of the prime motivations for compression software.

Batch Job

A predefined group of processing actions submitted to the system to be performed with little or no interaction between you and the system. This is in contrast to an Interactive Job.

Big ENDIAN

A binary data format in which the most significant bit comes first.

Binary File

A file that contains codes that are not part of the ASCII character set. Binary files can use all 256 possible values for each byte in the file.

Bit

A contraction of binary digit. Either of the binary digits, 0 or 1. Compare with byte.

Block

(1) A group of records that are recorded or processed as a unit.
(2) A set of adjacent records stored as a unit on a disk, diskette, or magnetic tape.

Byte

(1) The smallest unit of storage that can be addressed directly.
(2) A group of 8 adjacent bits. In the EBCDIC/ASCII coding system, 1byte can represent a character. In the double-byte coding system, 2 bytes represent a character.

CBC

Cipher Block Chaining refers to a method of encryption of blocks of data that involves an initialization vector that is put together with the first block of data and the encryption key. This method of encryption makes sure that each block of data thereafter is uniquely modified, further protecting the data from fraudulent access.

CICS

See Customer Information Control System.

COBOL

See Common Business Oriented Language shown below.

Code Page

A specification of code points for each graphic character set or for a collection of graphic character sets. Within a given code page, a code point can have only one specific meaning. A code page is also sometimes known as a code set.

Common Business Oriented Language (COBOL)

A high-level programming language, based on English, that is used primarily for commercial data processing.

Command Line

The blank line on a display console where commands, option numbers, or selections can be entered.

Configuration File

(1) A file that specifies the way a program functions.
(2) In PKZIP, the file that contains the default values needed for the system to run. These can usually be respecified to meet local user requirements.

Control Language (CL) Program

A program that is created from source statements consisting entirely of control language commands.

CPU

The Central Processing Unit or processor in the computer is the hardware that allows the computer to process.

CRC

See Cyclic Redundancy Check.

Cryptography

(1) A method of protecting data. Cryptographic services include data encryption and message authentication.
(2) In cryptographic software, the transformation of data to conceal its meaning; secret code.
(3) The transformation of data to conceal its information content, to prevent its undetected modification, or to prevent its unauthorized use.

Customer Information Control System (CICS)

An IBM licensed program that enables transactions entered at remote workstations to be processed concurrently by user-written application programs. The licensed program includes functions for building, using, and maintaining databases, and forcommunicating with CICS programs on other operating systems.

Cyclic Redundancy Check (CRC)

A Cyclic Redundancy Check is a number derived from a block of data, and stored or transmitted with the data in order to detect any errors in transmission. This can also be used to check the contents of a ZIP archive. It is similar in nature to a checksum. A CRC may be calculated by adding words or bytes of the

data. Once the data arrives at the receiving computer, a calculation and comparison is made to the value originally transmitted. If the calculated values are different, a transmission error is indicated. The CRC information is called redundant because it adds no significant information to the transmission or archive itself. It is only used to check that the contents of a ZIP archive are correct. When a file is compressed, the CRC is calculated and a value is calculated based upon the contents and using a standard algorithm. The resulting value (32 bits in length) is the CRC that is stored with that compressed file. When the file is decompressed, the CRC is recalculated (again, based upon the extracted contents), and compared to the original CRC. Error results will be generated showing any file corruption that may have occurred.

Cylinder

The tracks on a disk or diskette that can be accessed without movement of the read/write arm and head.

DASD

Direct Access Storage Device is a peripheral device for a mainframe computer, like a disk or drum, that can be directly addressed.

Data Compression

The reduction in size (or space taken) of data volume on the media when performing a save or store operations.

Data Integrity

(1) The condition that exists as long as accidental or intentional destruction, alteration, or loss of data does not occur.
(2) Within the scope of a unit of work, either all changes to the database management systems are completed or none of them are. The set of change operations are considered an integral set.

DBCS

See Double-byte Character Set.

Delimiter

A character or sequence of characters that marks the beginning or end of a unit of data.

Device

A piece of equipment that is used with the computer. A device does not generally interact directly with the system, but is controlled by a controller.

Direct Access

A file access method allowing reading and writing of records in an arbitrary order. Contrast with keyed access and sequential access.

DOS

The Disk Operating System is an older operating system that used a manual disk swapping method to store, manipulate, and retrieve data.

Double-byte Character Set (DBCS)

A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, displaying, and printing of DBCS characters requires hardware and programs that support DBCS. Four double-byte character sets are supported by the system: Japanese, Korean, Simplified Chinese, and Traditional Chinese. See also the Single-Byte Character Set (SBCS).

Dump

In problem analysis and resolution, to write, at a particular instant, all or part of the contents of main or auxiliary storage onto another data medium (such as tape, printer, or spool) for the purpose of protecting the data or collecting error information.

DYNALLOC

See Dynamic Allocation shown below.

Dynamic Allocation

Dynamic Allocation (DYNALLOC) is a facility utilizing the SVC99 function which allows a program to directly access a dataset without the need for corresponding JCL statements.

EBCDIC

See the Extended Binary Coded Decimal Interchange Code shown below.

Encryption

The transformation of data into an unintelligible form so that the original data either cannot be obtained or can be obtained only by decryption.

ENQ

See the Enqueue macro shown below.

Enqueue

The Enqueue macro (ENQ) is used to restrict access to a resource, so that only the appropriate number of users with the appropriate mode gain access to the resource at one time. It is commonly used to "lock" a resource to prevent modifications from multiple sources to cancel out each other.

ESA

Enterprise Systems Architecture is IBM enhancements for mainframe computers to increase virtual memory and manage it effeiciently.

Extended Attribute

Information attached to an object that provides a detailed description about the object to an application system or user.

Extended Binary Coded Decimal Interchange Code (EBCDIC)

The Extended Binary Coded Decimal Interchange Code a coded character set of 256 8 bit characters. EBCDIC is similar in nature to ASCII code, which is used on many other computers. When ZIP programs compress a text file, they translate data from EBCDIC to ASCII characters within a ZIP archive using a translation table.

FAQ

The Frequently Asked Questions is a repository of the questions most commonly asked by a user community. These question are generally posted on the company web site and updated periodically.

Fixed-Length

A dataset or data definition characteristic in which all of the records are the same length. See also Variable Length.

GDG

Generation Data Groups.

GMT

See Greenwich mean time shown below or Universal Time Coordinated (UTC).

GNU

A recursive acronym for the name of the Free Software Foundation's freely distributable replacement for UNIX.

Greenwich mean time (GMT)

A synonym for Universal Time Coordinated (UTC) which is the mean solar time of the meridian of Greenwich, England, and is the prime basis of standard time throughout the world.

GZIP

GZIP (also known as GNU zip) is a compression utility designed to use a different standard for handling compressed file data in an Archive.

Host

The controlling or highest-level system in a data communications configuration; for example, an iSeries system is the host system for the work stations connected to it.

ICF

Integrated Catalog Facility.

I/O

See Input/Output shown below.

IDCAMS

The utility program used by IBM’s Access Method Services to create and manage VSAM datasets.

Input/Output (I/O)

Data provided to the computer or data resulting from computer processing.

Installation Verification Procedure (IVP)

A sample application, script, or jobstream provided to verify successful installation of a product (may be either software or hardware).

IO

See Input/Output.

iSeries

See Application System/400.

IVP

See Installation Verification Procedure.

JCL

Job Control Language is a command language for mainframes and minicomputers, used for launching applications.

JES2 or JES3

See Job Entry Subsystem shown below.

Job Entry Subsystem (JES)

A System/370-type licensed program that receives jobs into the system and processes all output data produced by the jobs. Commonly known as JES2, JES3, or POWER.

Julian Date

A date format that contains the year in positions 1 and 2, and the day in positions 3 through 5. The day is represented as 1 through 366, right-adjusted, with zeros in the unused high-order positions. For example, the Julian date for April 6, 1987 is 87096.

Kanji

Characters originating from the Chinese characters used in the Japanese written language.

Keyed Sequence

An order in which records are retrieved based on the contents of key fields in records. For example, a bank name and address file might be in order and keyed by the account number.

Keyword

(1) A mnemonic (abbreviation) that identifies a parameter in a command.
(2) A user-defined word used as one of the search values to identify a document during a search operation.
(3) In COBOL, a reserved word that is required by the syntax of a COBOL statement or entry.
(4) In DDS, a name that identifies a function.
(5) In REXX, a symbol reserved for use by the language processor in a certain context. Keywords include the names of the instructions and ELSE, END, OTHERWISE, THEN, and WHEN.
(6) In query management, one of the predefined words associated with a query command.
(7) A name that identifies a parameter used in an SQL statement. Also see parameter.

Label

(1) The name of a file on a diskette or tape.
(2) An identifier within a command or program statement generally used for branching.
(3) In Interactive Source Debugger, a place in a source program that you can choose to display again.
(4) In REXX, a clause that consists of a single symbol followed by a colon.
(5) In RPG, a symbolic name that represents a specific location in a program. A label can serve as the destination point for one or more branching operations.
(6) In DB2 UDB for iSeries SQL, a way of attaching text to columns, tables, and packages.
(7) In Backup Recovery and Media Services, an external identifier for media. A label includes information about volume serial identifier, creation date, expiration date, location, and container identifier.

Lempel-Ziv (LZ)

A technique for compressing data. This technique replaces some character strings, which occur repeatedly within the data, with codes. The encoded character strings are then kept in a common dictionary, which is created as the data is being sent.

License Authorization Code (LAC)

The inserted code that is needed to unlock an iSeries licensed program.

Linkage Editor

A system-related program that resolves cross-references between separately compiled object modules and then assigns final storage addresses to create a single load module.

Little ENDIAN

A binary data format in which the least signifcant bit would be on the left.

LZ

See Lempel-Ziv (LZ).

Magnetic Tape Unit

See Tape Drive.

Member

Different groupings of data within one database file (such as members within a partitioned dataset).

MVS

Multiple Virtual Storage is the most commonly-used operating system on IBM mainframes; other operating systems are VM and DOS/VSE.

New ZIP Archive

A New ZIP archive is the archive created by a compression program when either an old ZIP archive is updated or when files are compressed and no ZIP archive currently exists. It may be thought of as the "receiving" archive. Also see Old ZIP Archive shown below.

NIST

National Institute of Standards and Technology is a part of the U.S. Department of Commerce, formerly called the National Bureau of Standards, that defines standards for voice, data, and video transmissions, encryption, and other kinds of technology.

Null Value

A parameter which has no value assigned.

Old ZIP Archive

An Old ZIP archive is an existing archive which is opened by a compression program to be updated or for its contents to be extracted. It may be thought of as the "sending" archive. Also see New ZIP Archive.

Operating System/400 (OS/400)

An IBM-licensed program that is as the primary operating system for an iSeries system.

OS/400

See Operating System/400 shown above.

Output

Information or data received from a computer that is shown on a display, printed on the printer, communicated, or stored on disk, diskette, or tape.

Packed Decimal Format

A decimal value in which each byte within a field represents two numeric digits except the far right byte, which contains one digit in bits 0 through 3 and the sign in bits 4 through 7. For all other bytes, bits 0 through 3 represent one digit; bits 4 through 7 represent one digit. For example, the decimal value +123 is represented as 0001 0010 0011 1111 (or 123F in hexadecimal).

Parameter

(1) A value supplied to a command or program that is used either as input or to control the actions of the command or program.
(2) In COBOL, a variable or a constant that is used to pass values between calling and called programs.
(3) In the Integrated Language Environment (ILE), an identifier that defines the types of arguments that are passed to a called procedure.
(4) In REXX, information entered with a command name to define the data on which a command processor operates and to control the execution of the command.
(5) In DB2 UDB for iSeries SQL, the keywords and values that further define SQL precompiler commands and SQL statements. Also see keyword.

Parameter List

A list of values in a calling program that corresponds exactly to a list in a called program for the purposes of providing addressability and data exchange. It contains parameter names and the order in which they are to be associated in the calling and called program.

Partition

A fixed-size portion of the available storage.

Partitioned Dataset

A Partitioned Dataset (PDS) is a dataset in direct access storage that is divided into partitions (which are called members), each of which can contain a program, part of a program, JCL, parameters, or other forms of data. When a compression program is compressing a PDS, each member is treated as a separate file within the resultant ZIP archive. When an archive is decompressed to a PDS, each file within the archive creates a separate member within the PDS.

Path Name

(1) A string of characters used to refer to an object. The string can consist of one or more elements, each separated by a slash (/), and may begin with a slash. Each element is typically a directory or equivalent, except for the last element, which can be a directory or another object (such as a file).
(2) A sequence of directory names followed by a file name, each separated by a slash.

PF or PFK

See Program Function Key.

PDS

See Partitioned Dataset.

PL/I

See Programming Language/I.

POWER

See Job Entry Subsystem.

Production Library

A library which contains objects needed for normal processing. This contrasts with a Test Library.

Programmed Function Key (PF or PFK)

On a workstation, a specified series of keys that can perform various functions selected by you or determined by an application program.

Programming Language/I (PL/I)

A programming language designed for use in a wide range of commercial and scientific computer applications.

Program Temporary Fix (PTF)

A temporary solution to (or a bypass of) a problem that is necessary to provide a complete solution to correct a defect in a current unaltered release of a program. May also be used to provide an enhancement to a product before a new release of the product is available. Generally, PTFs are incorporated in a future release of the product.

PSW

Program Status Word.

PTF

See Program Temporary Fix.

RDW

Record Descriptor Word.

Record

A group of related data, words, or fields treated as a single unit, such as a name, address, and social security number.

Record Format

A document or display that names each part of a file and provides specific information for each field such as length and type of information contained within the field.

Relative Path Name

A string of characters that is used to refer to an object, starting at some point in the directory hierarchy other than the root. A relative path name does not begin with a slash (/). The starting point is frequently a user's current directory. This is in contrast to an Absolute Path Name and Path Name.

Return Code

A value generated by operating system software to a program to indicate the results of an operation by that program. The value may also be generated by the program and passed back to the operator.

Report Program Generator (RPG)

A programming language used on iSeries and mainframes for writing application programs for business data processing requirements.

Rijindael

The combined name of the two researchers that developed the Advanced Encryption Standard (AES) for the US Government (Dr. Joan Daemen and Dr. Vincent Rijmen).

RPG

See Report Program Generator shown above.

SBCS

See Single-Byte Character Set.

SEQ

See Sequential Dataset.

Sequential Dataset

A sequential dataset holds a single file of records which are organized on the basis of their successive physical positions, such as on magnetic tape.

Single-Byte Character Set (SBCS)

A coded character set in which each character is represented by a one-byte code point. A one-byte code point allows representation of up to 256 characters. Languages that are based on an alphabet, such as the Latin alphabet (as contrasted with languages that are based on ideographic characters) are usually represented by a single-byte coded character set. For example, the Spanish language can be represented by a single-byte coded character set. Also see the Double-Byte Character Set (DBCS).

Source File

A file of programming code that has not yet been compiled into machine language. A source file can be created by the specification of FILETYPE(*SRC) on the Create command. A source file can contain source statements for such items as high-level language programs and data description specifications. Source files maintained on a PC typically use a .TXT as the extension. On a mainframe, source files are typically found in a partitioned dataset or are maintained within a library management tool.

Spanned Record

A logical record that stored across more than one block. This is commonly used to get around system limitations that blocks cannot be larger than x number of bytes. With spanned records, one record spans two or more blocks.

Tape Cartridge

A formed case containing a small reel of magnetic tape that can be put into a tape drive without stringing the tape between reels.

Tape Drive

A hardware device that is used to read and write information on magnetic tapes or cartridges.

Tape Volume

A single reel of magnetic tape. May also be used to describe a single tape cartridge.

Test Library

A user-defined library used for debugging software or operations. This contrasts with a Production Library.

Time-Sharing Option

Interactive processing software used on the System/370 operating system for remote terminals.

Time Stamp

A software mechanism for recording the current date and/or current time of day.

Translation Table

Translation tables are used by the PKZIP and PKUNZIP programs for translating characters in compressed text files between the ASCII character sets used within a ZIP archive and the EBCDIC character set used on IBM-based systems. These tables may be created and modified by you as documented in the user's guide.

Truncate

To cut off or delete the data that will not fit within a specified line width or display. This may also be attributed to data that does not fit within the specified length of a field definition.

Unit

The defined space within designated disk units that is addressed by the operating system.

Universal Time Coordinated (UTC)

A synonym for Greenwich Mean Time (GMT) which is the mean solar time of the meridian of Greenwich, England, and is the prime basis of standard time throughout the world.

User Interface

The actions or items that allow you to interact with (and/or perform operations on) a computer.

UTC

See Universal Time Coordinated (UTC) (shown above) or Greenwich mean time.

Variable-Length

A characteristic of a file in which the individual records (and/or the file itself) can be of varying length. Also see Fixed-Length.

Virtual Storage Access Method

The Virtual Sequential Access Method (VSAM) is an access method for the direct or sequential processing of fixed-length and variable-length records on direct access devices. The records in a VSAM

dataset or file can be organized in logical sequence by a key field (key sequence dataset or KSDS), in the physical sequence in which they are written on the dataset or file (entry-sequence or PS), or by relative-record number (RR). The datasets are managed by the IDCAMS utility program and is used by commands and macros from within application programs.

VTOC

See Volume Table of Contents.

Volume

A storage device that can be taken off the system as an individual unit, for example, magnetic tape, disk, or diskette.

Volume Label

A standard tape or cartridge contains a VOL1 label as the first record of the tape (80 bytes with VOL1 beginning in the first position). The label is used to identify the tape volume and its owner.

Volume Table of Contents (VTOC)

An area on a disk or diskette that contains descriptions, for example, location, size, and other characteristics of the files, libraries, and folders that exist on the disk or diskette.

VSAM

See Virtual Storage Access Method.

ZIP Archive

A ZIP archive is used to refer to a single dataset that contains a number of files compressed into a much smaller physical space by PKZIP software.

Index

$

$INSTLIC, 18

&

&SYSUID, 62

A

A, 164
About this Manual, iii
Absolute Path Name, 200
Access Method, 200
Access Method Services, 13
ACTION, 62, 76
-ACTION(VIEWDETAIL), 25
Activating the ISPF Interface, 19
ACZDFLT, 18
–ADD, 76
Advanced Options, 167
AES Key Sizes, 5
–ALIAS_NAME, 118
–ALIASMEMBER, 115
Allocation Units, 166
Alternate Index, 200
American National Standards Institute, 200
American Standard Code for Information Interchange, 200
ANSI, 200
APAR, 200
API, 171, 175, 201
Appendix A - Licensing Requirements, 174
Appendix B - Sample Jobstreams, 178
Appendix C - Invoking PKZIP/PKUNZIP from a Program, 188
Appendix D - 3480/3490 Installation JCL (COPYCART), 195
Application Programming Interface, 201
Application Programming Interface (API), 171
Application System/400 (iSeries), 201
Applying a License Key or Authorization Code, 17
–ARCH, 149
–ARCHBLKSIZ, 79
–ARCHBUFSPACE, 130
–ARCHCATALOG, 130
–ARCHCISIZE, 131
–ARCHCISZ, 131
–ARCHDATACISIZE, 132
–ARCHDATACISZ, 132
–ARCHDATAEEXT, 132
–ARCHDATAFILE, 133
–ARCHDATANAME, 133
–ARCHDATANORD, 133
–ARCHDATANRUS, 146
–ARCHDATANWCK, 149
–ARCHDATAORD, 133
–ARCHDATAOWNER, 144
–ARCHDATAPRI, 134
–ARCHDATARUS, 146
–ARCHDATASEC, 134
–ARCHDATASHR, 146
–ARCHDATASPACE, 134
–ARCHDATAVOL, 135
–ARCHDCLASS, 80
–ARCHDIRBLKS, 80
–ARCHEEXT, 136
–ARCHERASE, 135
–ARCHFILE, 136
–ARCHFOR, 136, 148
–ARCHFREECA, 137
–ARCHFREECI, 137
–ARCHIFILE, 82
–ARCHINDD, 82
–ARCHINFILE, 82
Archive, 201
–ARCHIVE, 80
Archive Name, 161, 166
ARCHIVE_BLKSIZE, 62, 79
ARCHIVE_COMMENT, 63, 79
ARCHIVE_DATACLASS, 63, 80
ARCHIVE_DIR_BLOCKS, 63, 80
–ARCHIVE_DIRBLKS, 80
ARCHIVE_DSN, 63, 80
–ARCHIVE_DSNAME, 80
ARCHIVE_DSORG, 63, 81
–ARCHIVE_IFILE, 82
–ARCHIVE_INDD, 82
ARCHIVE_INFILE, 63, 82
ARCHIVE_LRECL, 63, 82
ARCHIVE_MGMTCLASS, 63, 82
–ARCHIVE_MODEL, 144
–ARCHIVE_OFILE, 82
–ARCHIVE_OUTDD, 82
ARCHIVE_OUTFILE, 63, 82
ARCHIVE_RECFM, 63, 83
–ARCHIVE_RELEASE, 83
–ARCHIVE_RLSE, 83
ARCHIVE_SPACE_PRIMARY, 63, 83
–ARCHIVE_SPACE_RELEASE, 83
ARCHIVE_SPACE_RLSE, 63, 83

ARCHIVE_SPACE_SECONDARY, 63, 84
ARCHIVE_SPACE_TYPE, 63, 84
ARCHIVE_STORCLASS, 19, 63, 84
ARCHIVE_TIMESTAMP, 63, 85
ARCHIVE_UNIT, 19, 63, 85
ARCHIVE_VOLUMES, 19, 63, 85
–ARCHLRL, 82
–ARCHMCLASS, 82
–ARCHMODEL, 144
–ARCHNOERASE, 135
–ARCHNONSPANNED, 148
–ARCHNOREUSE, 146
–ARCHNORLSE, 83
–ARCHNOWRITECHK, 149
–ARCHOFILE, 82
–ARCHOUTDD, 82
–ARCHOUTFILE, 82
–ARCHOWNER, 144
–ARCHPRIMARY, 83
–ARCHRECORDSIZE, 145
–ARCHREUSE, 146
–ARCHRLSE, 83
–ARCHSCLASS, 84
–ARCHSECONDARY, 84
–ARCHSHR, 146
–ARCHSPACE, 84
–ARCHSPANNED, 148
–ARCHTO, 148
–ARCHTYPE, 83
–ARCHUNIT, 85
–ARCHVOL, 85
–ARCHWRITECHK, 149
ASCII, 201
ASMDFLT, 18
ASMSAFE, 19
Assembling Your Changes, 32
–ATTRCOMPAT, 86
–ATTRIB, 116
–ATTRIB_COMPAT, 86
ATTRIB_COMPATIBILITY, 63, 86
–ATTRIBCENTRAL, 116
–ATTRIBLOCAL, 116
–ATTRIBUTE_COMPATIBILITY, 86
Authorized Program Analysis Report (APAR), 201

B

B, 164
Bandwidth, 201
BASIC, 174
Batch Job, 201
BB, 164
Big ENDIAN, 201
–BINARY, 90
Binary File, 201
Binary Records, 44
Bit, 201
Block, 201
Block Size, 165
Browse, 164
Browse Binary, 164
Browse Text, 164
BT, 164
–BUFFERSPACE, 130
–BUFSPACE, 130
Byte, 201

C

–CACHEMEMORY, 90
CALLMODE, 63, 86
CANCEL, 159
CAPACITY, 174
–CATALOG, 130
Cataloged Dataset Name and INFILE Request Restrictions, 37
Cataloged Dataset Name Filter Requests, 35
Changing Default Options, 160
Chapter 1. An Introduction to PKZIP, 1
Chapter 1. - An Introduction to PKZIP MVS, 1
Chapter 10. - Processing with GZIP, 154
Chapter 11. - Using the ISPF Interface, 157
Chapter 12. - Using the Application Programming Interface (API), 171
Chapter 2. - PKZIP for MVS™ 5.5 Release Information, 7
Chapter 3. - Installation, Licensing, and Configuration, 14
Chapter 4. - Getting Started with *PKZIP for MVS*™, 21
Chapter 5. File Selection and Name Processing, 35
Chapter 6. - ZIP Files, 42
Chapter 7. - File Processing, 46
Chapter 8. - ZIP Archives, 59
Chapter 9. - Commands, 61
–CHECK_SYSIN_MEMBER, 63, 86
CICS, 202
–CISIZE, 131
CLI, 175
–CNVEXT, 97
COBOL, 202
Code Page, 202
Command Changes, 8
Command Details, 74
Command Icon Legend, 76
Command Line, 202
Command Syntax, 61
Commands, 61
Common Business Oriented Language (COBOL), 202

Compress and Store all of a User's Files into Their Own Archive, 30
Compressed by, 166
Compressing a Dataset, 23
Compressing a VSAM File, 51
Compressing Data from Tape, 55
Compressing Sequential Files, 47
COMPRESSION, 175
Compression Method, 166
Compression Ratio, 165
COMPRESSION_LEVEL, 63, 87
Conditional Use, 177
Configuration (Option 'C'), 158
Configuration File, 202
Configuration Manager, 31
Configuration Manager Development: Managing Control Statements, 33
Contact PKWARE (Option 'A'), 170
Contents, vii
Control Language (CL) Program, 202
Control Statement Definitions, 33
Conventions Used, iv
–COPY, 76
Copying a Tape-Based Archive to a Disk File, 54
Could "DES Cracker" like hardware break an AES key?, 5
CRC, 202
CREATED, 163
Creation Date, 166
CRLF, 63, 87
Cross Platform Compatibility, 6
Cryptography, 202
Current Use License, 175
Customer Information Control System (CICS), 202
Cyclic Redundancy Check, 2, 166
Cyclic Redundancy Check (CRC), 202
Cylinder, 203

D

D, 164
Data Compression, 1, 203
Data Encryption, 4
Data Format - Binary Records, 44
Data Format - Text Records, 43
Data Formats - Text or Binary, 42
Data Integrity, 203
Data Set Filter, 161
Data Set Name, 160
DATA_DELIMITER, 64, 89
DATA_STORAGE, 64, 90
DATA_TYPE, 64, 90
Alias, 10
Dataset Aliases, 10
dataset name, 62, 74
DATATYPE_DETECT_DEPTH, 92
DATATYPE_DETECT_TABLE, 92
–DATATYPE_SCAN_DEPTH, 92
DATATYPE_TEXT_PERCENT, 93
DATE, 163
Date/Time Zipped, 165
DBCS, 203
DD Statements, 49
DDNAME_PARMLIB, 64, 93
DDNAME_QZSORTIN, 64
DDNAME_QZSORTOUT, 64
DDNAME_SYSIN, 64, 93
DDNAME_SYSPRINT, 64, 94
DDNAME_ZPSORTIN, 94
DDNAME_ZPSORTOUT, 94
Debugging Controls, 34
Decompressing sequential datasets, 26
Decompressing a Dataset, 26
DECOMPRESSION, 175
Defaults (Options ZD and UD), 158
Defaults Files, 158
Defaults for –ZIPPED_DSN, 151
Defaults Module, 18, 158
Delete, 164
–DELETE, 76
–DELIM, 89
Delimiter, 203
DEMO, 174, 176
–DETECT_DEPTH, 92
–DETECTX, 90
Determining File Size, 45
Device, 203
Direct Access, 203
Directory Blocks, 166
DISASTER RECOVERY, 174
DISP, 159
Display Fields, 165
Double-byte Character Set (DBCS), 203
Dsorg, 165
Dump, 204
DYNALLOC, 204
Dynamic Allocation, 204

E

–E0, 87
EBCDIC, 160, 204
ECHO, 64, 94
–EN, 87
Encryption, 167, 204
–ENCRYPTION_METHOD, 95
ENCRYTPION_METHOD, 64
Enhancements for Secure Data, 9
ENQ, 204
Enqueue, 204

ENTERPRISE, 174
Environmental Execution Considerations, 11
–ES, 87
–ESDS, 131
Essentials for running PKZIP and PKUNZIP, 39
–EX, 87
Example
-VIEWDETAIL, 25
Example 1: Zip PDS to an Archive, 178
Example 2: Zip PDS to an Archive, 179
Example 3: Zip VSAM KSDS to an Archive, 180
Example 4: Summary View of a Dataset, 181
Example 5: Summary View of a Dataset, 182
Example 6: View with Detail of an Archive, 183
Example 7: Unzip an Archive to PDS, 185
Example 8: Unzip an Archive to PDS, 186
Example 9: Unzip an Archive to VSAM KSDS, 187
Examples
extracting data, 26
viewing archive contents, 24
–EXCLUDE(dsname mask), 64, 95
Exclusion Filter, 35
EXIT, 159
Extended Attribute, 204
Extended Binary Coded Decimal Interchange Code (EBCDIC), 204
Extract, 165
–EXTRACT, 76
Extract with overwrite, 165
EXTRACT_PREVIEW, 64, 96
Extracting Data into a PDS, 49
Extracting Data into a VSAMFfile, 52
Extracting Data onto Tape, 56
Extracting Records into a Sequential File, 47

F

–FAILONDUPKEYS, 135
FEATURES, 174
Features Distinctive to PKZIP for MVS, 2
File Attributes, 44, 55
File Concatenation for ZIP Processing, 48
File Considerations, 44
File Name, 165
File Name or File Mask, 49
File Processing, 46
File Selection and Name Processing, 35
File Selection Processing Notes, 36
File Selections vs. Commands, 62
File Support, 46
File to Compress, 167
File Type, 165, 167
FILE_BUSY_WAITTIME, 64, 96
FILE_EXTENSION, 64, 97
FILE_TERMINATOR, 64, 97
–FILEPROCERR, 104
–FILESELERR, 103
Fixed-Length, 204
Format, 167
–FRESHEN, 76
–FTRAN, 125

G

–GDGALL, 98
GDGALL_SUPPORT, 64, 98
General Purpose, 31
Getting Started with the ISPF Interface, 157
GIGA ZIP, 175
Glossary, 200
GMT, 204
Greenwich mean time (GMT), 205
GZIP, 64, 99, 205
GZIP Extensions, 155
GZIP Restrictions, 155
GZIP_SUFFIX, 64, 99

H

HIERARCHY, 64, 99
–HLQ, 125
Host, 205

I

I, 164
I/O, 205
IBM's Terminology Web Site, 200
IDCAMS, 205
IEBGENER, 13
–IFILE, 100
–IGNOREDUPKEYS, 135
Implementation Notes for GZIP, 155
Including Changed Defaults, 161
–INDD, 100
Index, 212
INFILE, 65, 100
INFILE Requests, 36
–INFILE_DD, 100
Info, 164
Input ZIP Archive Files, 36
Input/Output (I/O), 205
Inputs, 32
INSERT_MEMBER, 65, 100
–INSERTMEMBER, 100
Installation Overview, 14
Installation Verification Procedure (IVP), 205
Installing from 3480 or 3490 Tape, 15
Introduction to PKZIP for MVS, 21
Invoking *PKZIP for MVS*™ Services, 27
Invoking PKZIP or PKUNZIP Using JCL, 21

Invoking the PKZIP for MVS™ ISPF Panel Interface, 30
Invoking the PKZIP for MVS™ ZIP and UNZIP Services API, 31
Invoking the PKZIP or PKUNZIP Programs as Called Programs Under TSO, 27
Invoking the PKZIP or PKUNZIP Programs From JCL (Batch or Started Task), 27
Invoking ZIP or UNZIP TSO Command Line Interface, 28
Invoking ZIPAPI in Assembler, 171
Invoking ZIPAPI in COBOL, 172
IO, 205
iSeries, 205
ISPF, 175
ISPF Main Menu, 20
ISR@PRIM, 20
IVP, 205

J

JCL to run PKZIP, 21
JES2, 205
JES2 SYSIN INFILE Support, 36
JES3, 205
Job Card, 158
Job Entry Subsystem (JES), 205
JOBLIB DD, 22
Julian Date, 206

K

Kanji, 206
–KEY_PROTECT_LEVEL, 65, 101
Keyed Sequence, 206
–KEYPROTECT1, 101
–KEYPROTECT2, 101
Keyword, 206
–KSDS, –RRDS, 131

L

Label, 206
Large File Considerations, 44
Last-Referenced Date, 166
License Authorization Code (LAC), 206
License Display (Option 'L'), 169
LICENSE_HLQ, 18, 65, 101
Licensed Types, 174
Licensing, 46
Licensing and Initializing the Demo, 16
Licensing Environment, 175
Licensing Requirements, 174
LICPRINT, 176
LICSHSYS, 16, 177
LICxxxx, 18
Line Commands, 164
Linkage Editor, 206
Little ENDIAN, 206
–LMM, 102
LMOD_SUPPORT, 65, 101
LOAD, 159
Load Libraries, 50
Load Library, 158
Load Module Control, 50
LOCATE, 160
LOGGING_LEVEL, 65, 102
Lowest Acceptable RC, 158
LZ, 206

M

Magnetic Tape Unit, 206
MAGNETIC TAPE/CARTRIDGE FILE HANDLER, 175
Magnetic Tapes and Cartridges, 54
–MAKEESDS, 106
–MAKELIBRARY, 106
–MAKEPDS, 106
–MAKEPDSE, 106
–MAKESEQ, 106
–MAKEVSAM, 106
Making Changes to the Defaults, 31
Managing a Sequential File ZIP Archive, 48
Managing a VSAM ZIP Archive, 53
Managing a ZIP Archive on Tape, 56
Managing ZIP Archives as PDS Members, 50
Media Distribution for Installation, 14
–MEM_MDL, 102
–MEM_MODEL, 102
Member, 207
MEMORY_MODEL, 102
–MEMORY_MODEL, 65
Message, 165
Message Changes, 9
Messages, 33
Messages (Option 'M'), 168
–METHOD, 87
–MML, 102
–MMM, 102
–MMS, 102
Monitoring Algorithm Security, 5
More Attributes, 167
More Files, 167
MULTI_THREAD_LIMIT, 65, 103
–MULTI_THREAD_LIMIT, 65

N

NAME, 163
Needed to Extract, 166
New Commands, 7
New Features, 7
New ZIP Archive, 60, 207

–NIASEP, 152
–NOA, 125
–NOALIAS_NAME, 118
NOALIASMEMBER, 115
–NOARCHRLSE, 83
–NOATTRIB, 116
–NOCRLF, 87
–NODYNMSGS, 121
–NOECHO, 94
–NOGDGALL, 98
–NOGZIP, 99
–NOHIERARCHY, 99
–NOINSERTMEMBER, 100
Non-labeled Tapes (NL), 55
–NOOVERWRITE, 107
–NOPADVSAM, 111
–NOPATH, 113
–NORECALL, 115
–NORECURSE, 115
NOSYSIN, 65, 103
–NOSYSIPT, 103
–NOTAPE, 119
Notes for Dataset Compression, 23
Notes for Decompressing a Dataset, 26
Notes for invoking PKZIP or PKUNZIP using JCL, 22
Notes for Invoking PKZIP or PKUNZIP Using JCL, 22
Notes for Viewing the Contents of an Archive, 24
Notes for –ZIPPED_DSN, 150
–NOVSAM, 128
Null Value, 207
Numeric, 160

O

Old ZIP Archive, 59, 207
ON_FILE_ACCESS_ERROR, 65, 103
ON_FILE_IO_ERROR, 65, 104
Option ‘A’, 170
Option ‘C’, 158
Option ‘L’, 169
Option ‘M’, 168
Option ‘S’, 167
Option ‘V’, 161
Option ‘W’, 169
Option ‘Z’, 166
Option List, 160
Options ZD and UD, 158
OS/400, 207
–OUT_DSORG, 106
–OUTASTR, 129
–OUTATTEMPTS, 129
–OUTATTR, 131
–OUTAUTH, 129
–OUTBLKSIZ, 104
–OUTBLKSIZE, 104
–OUTBUFSPACE, 130
–OUTCATALOG, 130
–OUTCISIZE, 131
–OUTCISZ, 131
–OUTCODE, 131
–OUTCONTROLPW, 132
–OUTDATAASTR, 129
–OUTDATAATT, 129
–OUTDATAAUTH, 129
–OUTDATACISIZE, 132
–OUTDATACISZ, 132
–OUTDATACODE, 131
–OUTDATACTLPW, 132
–OUTDATAEEXT, 132
–OUTDATAFILE, 133
–OUTDATAMRPW, 143
–OUTDATANAME, 133
–OUTDATANORD, 133
–OUTDATANRUS, 146
–OUTDATANWCK, 149
–OUTDATAORD, 133
–OUTDATAOWNER, 144
–OUTDATAPRI, 134
–OUTDATARDPW, 145
–OUTDATARUS, 146
–OUTDATASEC, 134
–OUTDATASHR, 146
–OUTDATASPACE, 134
–OUTDATAUPDPW, 149
OUTDATAUSCL, 65
–OUTDATAVOL, 135
–OUTDATAWCK, 149
–OUTDCLASS, 105
–OUTDIRBLKS, 105
–OUTDUPLICATES, 135
–OUTEEXT, 136
OUTFILE_BLKSIZE, 65, 104
OUTFILE_DATACLASS, 65, 105
OUTFILE_DD, 65, 105
OUTFILE_DIR_BLOCKS, 65, 105
–OUTFILE_DIRBLKS, 105
OUTFILE_DSNTYPE, 65, 106
–OUTFILE_DSORG, 106
OUTFILE_LRECL, 65, 107
OUTFILE_MGMTCLASS, 65, 107
OUTFILE_OVERWRITE, 65, 107
OUTFILE_PDS_ENQ, 65, 107
OUTFILE_RECFM, 66, 108
–OUTFILE_RELEASE, 109
–OUTFILE_RLSE, 109
OUTFILE_SPACE_PRIMARY, 66, 109
–OUTFILE_SPACE_RELEASE, 109
OUTFILE_SPACE_RLSE, 66, 109
OUTFILE_SPACE_SECONDARY, 66, 109
OUTFILE_SPACE_TYPE, 66, 110

OUTFILE_STORCLASS, 19, 66, 110
OUTFILE_UNIT, 19, 66, 110
OUTFILE_VOLUMES, 19, 66, 110
–OUTFOR, 148
–OUTFREECA, 137
–OUTIMBED, 137
–OUTINDXASTR, 138
–OUTINDXATT, 137
–OUTINDXAUTH, 138
–OUTINDXCISIZE, 138
–OUTINDXCISZ, 138
–OUTINDXCTLPW, 139
–OUTINDXEEXT, 139
–OUTINDXNAME, 140
–OUTINDXNORD, 140
–OUTINDXNRUS, 146
–OUTINDXORD, 140
–OUTINDXOWNER, 144
–OUTINDXPRI, 141
–OUTINDXRDPW, 141
–OUTINDXRUS, 146
–OUTINDXSEC, 141
–OUTINDXSHR, 146
–OUTINDXSPACE, 142
–OUTINDXUPDPW, 142
OUTINDXUSCL, 66
–OUTINDXVOL, 143
–OUTKEYS, 143
–OUTLRL, 107
–OUTMASTERPW, 143
–OUTMCLASS, 107
–OUTMODEL, 144
–OUTNOREPLICATE, 146
–OUTNOREUSE, 146
–OUTNORLSE, 109
–OUTNOWRITECHK, 149
–OUTOWNER, 144
–OUTPRIMARY, 109
Output, 207
–OUTREADPW, 145
–OUTRECOVERY, 145
–OUTREPLICATE, 146
–OUTREUSE, 146
–OUTRLSE, 109
–OUTSCLASS, 110
–OUTSECONDARY, 109
–OUTSHR, 146
–OUTSPACE, 110
–OUTSPEED, 145
–OUTTO, 148
–OUTTYPE, 108
–OUTUNIT, 110
–OUTUPDATEPW, 149
OUTUSECLASS, 66
–OUTVOL, 110
–OUTWRITECHK, 149
–OVERWRITE, 107

P

Packed Decimal Format, 207
–PAD, 111
PAD_CHAR, 66, 111
PAD_VSAM, 66, 111
–PADVSAM, 111
Parameter, 207
Parameter List, 208
PARMLIB_DSNAME_UNZIP, 66, 111
PARMLIB_DSNAME_ZIP, 66, 112
PARMLIB_FILE_WAIT_MAX, 66, 112
PARMLIB_FILE_WAIT_TIMER, 66, 112
Partition, 208
Partitioned Data Set, 208
–PASS, 113
PASSWORD, 66, 113
PATCH_REPORT, 66
–PATCH_REPORT, 76
PATH, 66, 113
Path Name, 208
PDS, 208
PDS and PDSE Members, 48
PDS FILE HANDLER, 175
PDS/E FILE HANDLER, 175
–PDS_TARGET, 118
PF, 208
PFK, 208
PKNODUMP, 13
PKSPRINT, 13
–PKSUPPRC, 66, 114
PKUNZIP, 40
PKZALLOC, 20
PKZIP for MVS Grace Period, 17
PKZIPMVS.EXE, 14
PL/I, 208
POWER, 208
Preface, iii
–PRESERVE_CMD_SPACES, 66
–PREVIEW, 96
Preview Extract, 164
Primary Commands, 159, 163
Primary File Selection Inputs, 35
Primary Space, 165
PROCESS_ALIAS, 67, 115
Processing Entire Load Library, 50
Processing GDGs, 48
Processing GZIP Archives, 156
Processing Individual Members, 50
Processing Mode, 162, 167
Processing Order of Control Statements, 32
Processing with GZIP, 154
Product Features, 175
Production Library, 208

Program Temporary Fix (PTF), 208
Programmed Function Key (PF or PFK), 208
Programming Language/I (PL/I), 208
Protecting Files with the SAFETYEX Module, 19
PTF, 208
PV, 164
–PWD, 113

Q

–Q, 102
–QUIET, 102

R

RATIO, 163
–RDW, 117
–RECALL, 115
RECALL_TO_ZIP, 67, 115
Record, 209
Record Format, 165, 209
Record Size, 165
–RECURSE, 115
RECURSE_LEVELS, 67, 115
REF, 163
Region Size and Storage Usage, 11
Related IBM Publications, v
Related Publications, v
Relative Path Name, 209
Release Summary, 7
Report Program Generator (RPG), 209
Reporting, 176
Reporting the PKZIP for MVS 5.5 License, 16
Reserved DDNAMEs, 12
RESET, 160
Restriction for PKZIP Version 5.5 for MVS, 9
Return Code, 209
Return Codes, 22
RETURN CODES, 173
RPG, 209
Running a Disaster Recovery Test, 18

S

SAFETYEX, 19
SAFETYEX Module, 19
Sample Jobstreams, 178
SAVE, 160
SAVE_FILE_ATTRIBUTES, 67, 116
SAVE_LRECL, 67, 117
SBCS, 209
Secondary Space, 166
SELECT_CATALOGED_ALIAS, 67, 118
–SELECT_DSN_ALIAS, 118
SELECT_FROM_PDS, 67, 118
–SELECT_GDGALL, 98
–SELECT_MIGRATED, 115
SELECT_TAPE, 67, 119
–SELECT_VSAM, 128
Selecting PDS Members for Compression, 49
Self-Extracting ZIP File, 14
SEQ, 209
Sequential Dataset, 209
SEQUENTIAL FILE HANDLER, 175
Sequential Files, 47
SET_ERROR_RC, 67, 119
Setting VIEW Options, 161
Show System Information, 16, 177
SHOW_SETTINGS, 67, 119
SIMULATE, 67, 119
Simulation Mode, 167
–SMM, 102
SNAP_SYSOUT_CLASS, 67, 120
SORT, 13
Sort Field, 162
Sort Order, 162
Sort Output, 161
Source File, 209
Spanned Record, 210
Specific Changes, 18
–SS, 119
STAGE_TAPE_ON_DISK, 67, 120
–STAGE_TAPE_TO_DISK, 120
STEPLIB DD, 22
–STRIP, 120
STRIP_CHAR, 67, 120
Summary of Available Commands, 62
Summary of Commands Affecting ZIP Filename, 38
Summary View of a Dataset, 181, 182
SUPPRESS_DYNALLOC_MSGS, 67, 121
SYSEXEC, 20
SYSIN DD, 22
SYSPRINT, 12
Sysprint Allocation, 158
SYSPRINT Browse (Option ‘S’), 167
SYSPRINT DD, 22
SYSPRINT_DCB, 121
–SYSPRINT_DCB, 67
SYSPRINT_SYSOUT_CLASS, 67, 121
SYSPROC, 20

T

Tailoring Site Specific Changes, 18
Tape Cartridge, 210
Tape Drive, 210
Tape Volume, 210
TAPETODISK, 67
TASKS, 67
–TASKS, 103
TEMP_BLKSIZE, 67, 122
TEMP_DATACLASS, 67, 122

TEMP_MGMTCLASS, 68, 122
TEMP_RECFM, 68, 122
TEMP_SPACE_PRIMARY, 68, 123
TEMP_SPACE_SECONDARY, 68, 123
TEMP_SPACE_TYPE, 68, 123
TEMP_STORCLASS, 19, 68, 123
TEMP_UNIT, 19, 68, 123
TEMP_VOLUMES, 19, 68, 124
–TEMPBLKSIZ, 122
–TEMPDCLASS, 122
Temporary Dataset, 60
–TEMPPRI, 123
–TEMPPRIMARY, 123
–TEMPSCLASS, 123
–TEMPSEC, 123
–TEMPSECONDARY, 123
–TEMPSPACE, 123
–TEMPTYPE, 122
–TEMPUNIT, 123
–TERM, 97
–TEST, 76
Test Library, 210
Text, 160
–TEXT, 90
Text Records, 43
TIME, 163
Time Stamp, 210
TIME-DELIMITED, 174
Time-Sharing Option, 210
–TIMESTAMP, 85
To Compress Data into a ZIP Archive on Tape, 56
To Create a New VSAM File, 53
To Extract Data From a Tape-Based Archive, 57
To Overwrite a current VSAM File, 52
To Process “Sparse” RRDS Files, 54
To Process Multiple-Volume Tape Archives, 56
To Restore a Compressed VSAM File, 53
To Update a VSAM ESDS ZIP Archive, 54
To Update Files in a Tape-Based Archive, 58
To View a Tape-Based Archive, 57
TRACE_TABLE_SIZE, 68, 124
–TRAN, 124
TRANSLATE_TABLE_DATA, 68, 124
TRANSLATE_TABLE_FILEINFO, 68, 125
Translation Table, 210
TRANSLATION_MODE, 68
Trial Period, 16
Troubleshooting, 33
TRTEBAA, 125
TRTEBAI, 125
TRTEEAA, 125
TRTEEAI, 125
TRTEFAA, 125
TRTEFAI, 125
TRTEGAA, 125
TRTEGAI, 125
TRTEIAA, 125
TRTEIAI, 125
TRTEJAA, 125
TRTEJAI, 125
TRTEMAA, 125
TRTEMAI, 125
Truncate, 210
TSO Prefix, 158
Type of Media Distribution for Installation, 14

U

Unit, 210
Universal Time Coordinated (UTC), 210
Unsupported File Types, 54
Unzip an Archive to PDS, 185, 186
Unzip an Archive to VSAM KSDS, 187
–UNZIPCONFI, 111
Unzipped Size, 165
UNZIPPED_DSN, 68, 125
–UNZIPPED_DSNAME, 125
–UPDATE, 76
Updating or Refreshing a File, 27
Use of System Utilities, 13
USE_FILE_ATTRIBUTES, 116
–USE_SAVED_LRECL, 117
User Input Sources (MVS), 32
User Interface, 210
Using the ISPF Interface, 157
USIZE, 163
UTC, 210

V

V, 165
Valid UNZIP Actions, 29
Valid ZIP Actions, 28
Valid ZIP Options, 29
Variable-Length, 210
VB, 165
–VERBOSE, 102
View, 165
–VIEW, 76
View Archive (Option ‘V’), 161
View Binary, 165
View Text, 165
View Type, 161
View Typed Password, 167
View with Detail of an Archive, 183
–VIEWDETAIL Display, 55
VIEWDETAIL of a KSDS in an Archive, 51
Viewing the Contents of an Archive, 24
Virtual Storage Access Method, 210
Volume, 165, 211
Volume Label, 211
Volume List, 160
Volume Table of Contents (VTOC), 211

VSAM, 68, 128, 211
VSAM Clusters for –ZIPPED_DSN, 151, 152
VSAM FILE HANDLER, 175
VSAM Files, 50
VSAM_ACCOUNT, 68, 129
VSAM_ATTEMPTS, 68, 129
VSAM_AUTH_EP, 68, 129
VSAM_AUTH_STRING, 68, 129
VSAM_BUFFERSPACE, 68, 130
VSAM_CATALOG, 69, 130
VSAM_CISIZE, 69, 131
VSAM_CLUSTER_TYPE, 69, 131
VSAM_CODE, 69, 131
VSAM_CONTROLPW, 69, 132
VSAM_DATA_CISIZE, 69, 132
VSAM_DATA_EXCEPTIONEXIT, 69, 132
VSAM_DATA_FILE, 69, 133
VSAM_DATA_NAME, 69, 133
VSAM_DATA_ORDERED, 69, 133
VSAM_DATA_PRIMARY, 69, 134
VSAM_DATA_SECONDARY, 69, 134
VSAM_DATA_SPACE_TYPE, 69, 134
VSAM_DATA_VOLUMES, 69, 135
VSAM_DATACLASS, 69, 135
VSAM_DUPLICATE_ERROR, 70, 135
VSAM_ERASE, 70, 135
VSAM_EXCEPTIONEXIT, 70, 136
VSAM_FILE, 70, 136
VSAM_FOR, 70, 136
VSAM_FREESPACE_CA, 70, 137
VSAM_FREESPACE_CI, 70, 137
VSAM_IMBED, 70, 137
VSAM_INDEX_ATTEMPTS, 70, 137
VSAM_INDEX_AUTH_EP, 70, 138
VSAM_INDEX_AUTH_STRING, 70, 138
VSAM_INDEX_CISIZE, 70, 138
VSAM_INDEX_CODE, 70, 139
VSAM_INDEX_CONTROLPW, 70, 139
VSAM_INDEX_EXCEPTIONEXIT, 70, 139
VSAM_INDEX_FILE, 71, 140
VSAM_INDEX_MASTERPW, 71, 140
VSAM_INDEX_NAME, 71, 140
VSAM_INDEX_ORDERED, 71, 140
VSAM_INDEX_PRIMARY, 71, 141
VSAM_INDEX_READPW, 71, 141
VSAM_INDEX_SECONDARY, 71, 141
VSAM_INDEX_SPACE_TYPE, 71, 142
VSAM_INDEX_UPDATEPW, 71, 142
VSAM_INDEX_VOLUMES, 71, 143
VSAM_KEYS, 71, 143
VSAM_MASTERPW, 71, 143
VSAM_MGMTCLASS, 71, 144
VSAM_MODEL, 71, 144
VSAM_ORDERED, 71, 144
VSAM_OWNER, 72, 144
VSAM_READPW, 72, 145
VSAM_RECORDSIZE, 72, 145
VSAM_RECOVERY_OPT, 72, 145
VSAM_REPLICATE, 72, 146
VSAM_REUSE, 72, 146
VSAM_SHAREOPTIONS, 72, 146
–VSAM_SHROPT, 146
–VSAM_SHROPTS, 146
VSAM_SPACE_PRIMARY, 72, 147
VSAM_SPACE_SECONDARY, 72, 147
VSAM_SPACE_TYPE, 72, 147
VSAM_SPANNED, 72, 148
VSAM_STORCLASS, 19, 72, 148
VSAM_TO, 72, 148
–VSAM_TYPE, 131
VSAM_UPDATEPW, 72, 149
VSAM_USECLASS, 72
VSAM_VOLUMES, 19
–VSAM_VOLUMES, 135
VSAM_WRITECHECK, 72, 149
–VSAMCISIZE, 131
–VSAMCISZ, 131
–VSAMESDS, 131
–VSAMKSDS, 131
–VSAMRRDS, 131
–VSAMTYPE, 131
VT, 165
VTOC, 211

W

What is GZIP?, 154
What is the Life Expectancy of AES?, 6
What's New (Option 'W'), 169
Why use GZIP?, 154

X

X, 165
XO, 165

Y

Y/N, 160

Z

–ZDW, 117
Zip (Option 'Z'), 166
ZIP archive
  viewing contents, 24
ZIP Archive, 211
ZIP Archives, 1, 59
ZIP File Names, 38
ZIP Files, 42
Zip Function, 167
Zip PDS to an Archive, 178, 179
ZIP Processing File Selection, 35
Zip VSAM KSDS to an Archive, 180

–ZIPCONFIG, 112
–ZIPCUR, 118
ZIPPARM Copy Member, 172, 190, 192, 193, 194
Zipped DSN, 167
Zipped Size, 165
ZIPPED_DSN, 74, 149
ZIPPED_DSN_SEPARATOR, 74, 152
ZSIZE, 163